
EasyCV

Release 0.3.0

EasyCV Authors

Apr 29, 2022

USER GUIDE

1	Prepare Datasets	3
1.1	Prepare Cifar	3
1.2	Prepare Imagenet	3
1.3	Prepare Imagenet-TFrecords	4
1.4	Prepare COCO	4
1.5	Prepare PAI-Itag detection	4
2	Quick Start	5
2.1	Prerequisites	5
2.2	Installation	5
2.3	Examples	7
3	Self-supervised Learning Model Zoo	9
3.1	Pretrained models	9
3.2	Benchmarks	9
4	Detection Model Zoo	11
4.1	YOLOX	11
5	Develop	13
5.1	1. Code Style	13
5.2	2. Test	13
5.3	3. Build pip package	14
6	self-supervised learning tutorial	15
6.1	Data Preparation	15
6.2	Local & PAI-DSW	15
7	yolox tutorial	19
7.1	Data preparation	19
7.2	Local & PAI-DSW	19
8	image classification tutorial	23
8.1	Data Preparation	23
8.2	Local & PAI-DSW	24
9	file tutorial	27
9.1	Support operations	27
10	v 0.2.2 (07/04/2022)	35

11	easy cv.apis package	37
11.1	Submodules	37
11.2	easy cv.apis.export module	37
11.3	easy cv.apis.test module	37
11.4	easy cv.apis.train module	38
11.5	easy cv.apis.train_misc module	39
12	easy cv.datasets package	41
12.1	Subpackages	41
12.2	Submodules	131
12.3	easy cv.datasets.builder module	131
12.4	easy cv.datasets.registry module	131
13	easy cv.hooks package	133
13.1	Submodules	133
13.2	easy cv.hooks.best_ckpt_saver_hook module	133
13.3	easy cv.hooks.builder module	133
13.4	easy cv.hooks.byol_hook module	134
13.5	easy cv.hooks.dino_hook module	134
13.6	easy cv.hooks.ema_hook module	134
13.7	easy cv.hooks.eval_hook module	135
13.8	easy cv.hooks.export_hook module	136
13.9	easy cv.hooks.extractor module	137
13.10	easy cv.hooks.optimizer_hook module	137
13.11	easy cv.hooks.oss_sync_hook module	137
13.12	easy cv.hooks.registry module	138
13.13	easy cv.hooks.show_time_hook module	138
13.14	easy cv.hooks.swav_hook module	138
13.15	easy cv.hooks.sync_norm_hook module	139
13.16	easy cv.hooks.sync_random_size_hook module	139
13.17	easy cv.hooks.tensorboard module	140
13.18	easy cv.hooks.wandb module	140
13.19	easy cv.hooks.yolox_lr_hook module	140
13.20	easy cv.hooks.yolox_mode_switch_hook module	141
14	easy cv.predictors package	143
14.1	Submodules	143
14.2	easy cv.predictors.base module	143
14.3	easy cv.predictors.builder module	143
14.4	easy cv.predictors.classifier module	144
14.5	easy cv.predictors.detector module	145
14.6	easy cv.predictors.feature_extractor module	147
14.7	easy cv.predictors.interface module	150
14.8	easy cv.predictors.pose_predictor module	152
15	easy cv.core package	155
15.1	Subpackages	155
15.2	Submodules	184
15.3	easy cv.core.standard_fields module	184
16	easy cv.models package	191
16.1	Subpackages	191
16.2	Submodules	284
16.3	easy cv.models.base module	284
16.4	easy cv.models.builder module	285

16.5	easy cv.models.modelzoo module	285
16.6	easy cv.models.registry module	285
17	easy cv.utils package	287
17.1	Submodules	287
17.2	easy cv.utils.alias_multinomial module	287
17.3	easy cv.utils.bbox_util module	287
17.4	easy cv.utils.checkpoint module	288
17.5	easy cv.utils.collect module	289
17.6	easy cv.utils.collect_env module	289
17.7	easy cv.utils.config_tools module	289
17.8	easy cv.utils.constant module	290
17.9	easy cv.utils.dist_utils module	290
17.10	easy cv.utils.eval_utils module	291
17.11	easy cv.utils.flops_counter module	291
17.12	easy cv.utils.gather module	292
17.13	easy cv.utils.json_utils module	292
17.14	easy cv.utils.logger module	294
17.15	easy cv.utils.metric_distance module	294
17.16	easy cv.utils.misc module	295
17.17	easy cv.utils.preprocess_function module	295
17.18	easy cv.utils.profiling module	295
17.19	easy cv.utils.py_util module	296
17.20	easy cv.utils.registry module	296
17.21	easy cv.utils.test_util module	296
17.22	easy cv.utils.user_config_params_utils module	297
18	easy cv package	299
18.1	Subpackages	299
18.2	Submodules	307
18.3	easy cv.version module	307
19	easy cv	309
20	Indices and tables	311
	Python Module Index	313
	Index	317

EasyCV is an all-in-one computer vision toolbox based on PyTorch, mainly focus on self-supervised learning, image classification, metric-learning, object detection and so on.

PREPARE DATASETS

- [Prepare Cifar](#Prepare Cifar)
- [Prepare Imagenet](#Prepare Imagenet)
- [Prepare Imagenet-TFrecords](#Prepare Imagenet-TFrecords)
- [Prepare COCO](#Prepare COCO)
- [Prepare PAI-Itag detection](#Prepare PAI-Itag detection)

1.1 Prepare Cifar

Download dataset [cifar10](#) and uncompress files to `data/cifar`, directory structure is as follows:

```
data/cifar
├── cifar-10-batches-py
│   ├── batches.meta
│   ├── data_batch_1
│   ├── data_batch_2
│   ├── data_batch_3
│   ├── data_batch_4
│   ├── data_batch_5
│   ├── readme.html
│   ├── read.py
│   └── test_batch
```

1.2 Prepare Imagenet

1. Go to the [download-url](#), Register an account and log in .
2. Download the following files
 - Training images (Task 1 & 2). 138GB.
 - Validation images (all tasks). 6.3GB.
3. Unzip the downloaded file.
4. Using this [scrip](#) to get data meta.

1.3 Prepare Imagenet-TFrecords

1. Go to the [download-url](#), Register an account and log in .
2. The dataset is divided into two parts, [part0](#) (79GB) and [part1](#) (75GB), you need download all of them.

1.4 Prepare COCO

Download [COCO2017](#) dataset to data/coco, directory structure is as follows

```
data/coco
├── annotations
├── train2017
└── val2017
```

1.5 Prepare PAI-Itag detection

Download [SmallCOCO](#) dataset to data/coco, directory structure is as follows:

```
data/coco/
├── train2017
├── train2017_20_local.manifest
├── val2017
└── val2017_20_local.manifest
```

replace train_data and val_data path in config file

```
sed -i 's#train2017.manifest#train2017_20_local.manifest#g' configs/detection/yolox_coco_
↪pai.py
sed -i 's#val2017.manifest#val2017_20_local.manifest#g' configs/detection/yolox_coco_pai.
↪py
```

QUICK START

2.1 Prerequisites

- python \geq 3.6
- Pytorch \geq 1.5
- mmcv \geq 1.2.0
- nvidia-dali \geq 0.25.0

2.2 Installation

2.2.1 Prepare environment

1. Create a conda virtual environment and activate it.

```
conda create -n ev python=3.6 -y
conda activate ev
```

2. Install PyTorch and torchvision

The master branch works with **PyTorch 1.5.1** or higher.

```
conda install pytorch==1.7.0 torchvision==0.8.0 -c pytorch
```

3. Install some python dependencies

replace {cu_version} and {torch_version} to the version used in your environment

```
# install mmcv
pip install mmcv-full==1.4.4 -f https://download.openmmlab.com/mmcv/dist/{cu_
↪ version}/{torch_version}/index.html
# for example, install mmcv-full for cuda10.1 and pytorch 1.7.0
pip install mmcv-full==1.4.4 -f https://download.openmmlab.com/mmcv/dist/cu101/
↪ torch1.7.0/index.html

# install nvidia-dali
pip install http://pai-vision-data-hz.cn-hangzhou.oss-cdn.aliyun-inc.com/third_
↪ party/nvidia_dali_cuda100-0.25.0-1535750-py3-none-manylinux2014_x86_64.whl
```

(continues on next page)

(continued from previous page)

```
# install common_io for MaxCompute table read (optional)
pip install https://tfsmoke1.oss-cn-zhangjiakou.aliyuncs.com/tunnel_paiio/common_io/
↳py3/common_io-0.1.0-cp36-cp36m-linux_x86_64.whl
```

4. Install EasyCV

You can simply install easycv with the following command:

```
pip install pai-easycv
```

or clone the repository and then install it:

```
git clone https://github.com/Alibaba/EasyCV.git
cd easycv
pip install -r requirements.txt
pip install -v -e . # or "python setup.py develop"
```

5. Install pai_nni and blade_compression

When you use model quantize and prune, you need to install pai_nni and blade_compression with the following command:

```
# install torch >= 1.8.0
pip install torch==1.8.0 torchvision==0.9.0 torchaudio==0.8.0

# install mmcv >= 1.3.0 (torch version >= 1.8.0 does not support mmcv version < 1.3.
↳0)
pip install mmcv-full==1.4.4 -f https://download.openmmlab.com/mmcv/dist/{cu_
↳version}/{torch_version}/index.html

# install onnx and pai_nni
pip install onnx
pip install https://pai-nni.oss-cn-zhangjiakou.aliyuncs.com/release/2.5/pai_nni-2.5-
↳py3-none-manylinux1_x86_64.whl

# install blade_compression
pip install http://pai-vision-data-hz.oss-cn-zhangjiakou.aliyuncs.com/third_party/
↳blade_compression-0.0.1-py3-none-any.whl
```

2.2.2 Verification

Simple verification

```
```python
from easycv.apis import *
```
```

You can also verify your installation using following quick-start examples

2.3 Examples

- *Image classification example*
- *Self-supervised learning example*
- *object detection example*

SELF-SUPERVISED LEARNING MODEL ZOO

3.1 Pretrained models

3.1.1 MAE

Pretrained on **ImageNet** dataset.

3.1.2 DINO

Pretrained on **ImageNet** dataset.

3.1.3 MoBY

Pretrained on **ImageNet** dataset.

3.1.4 MoCo V2

Pretrained on **ImageNet** dataset.

3.1.5 SwAV

Pretrained on **ImageNet** dataset.

3.2 Benchmarks

For detailed usage of benchmark tools, please refer to benchmark [README.md](#).

3.2.1 ImageNet Linear Evaluation

3.2.2 ImageNet Finetuning

DETECTION MODEL ZOO

4.1 YOLOX

Pretrained on COCO2017 dataset.

5.1 1. Code Style

We adopt [PEP8](#) as the preferred code style.

We use the following tools: [flake8](#) for linting and [isort](#) for formatting:

- [flake8](#): linter
- [yapf](#): formatter
- [isort](#): sort imports

Style configurations of [yapf](#) and [isort](#) can be found in [setup.cfg](#). We use [pre-commit](#) hook that checks and formats for [flake8](#), [yapf](#), [seed-isort-config](#), [isort](#), [trailing whitespaces](#), [fixes end-of-files](#), [sorts requirements.txt](#) automatically on every commit. The config for a pre-commit hook is stored in [.pre-commit-config](#). After you clone the repository, you will need to install initialize pre-commit hook.

```
pip install -r requirements/tests.txt
```

From the repository folder

```
pre-commit install
```

After this on every commit check code linters and formatter will be enforced.

If you want to use pre-commit to check all the files, you can run

```
pre-commit run --all-files
```

If you only want to format and lint your code, you can run

```
sh scripts/linter.sh
```

5.2 2. Test

5.2.1 2.1 Unit test

```
bash scripts/ci_test.sh
```

5.2.2 2.2 Test data

if you add new data, please do the following to commit it to git-lfs before “git commit”:

```
python git-lfs/git_lfs.py add data/test/new_data
python git-lfs/git_lfs.py push
```

5.3 3. Build pip package

```
python setup.py sdist bdist_wheel
```

SELF-SUPERVISED LEARNING TUTORIAL

6.1 Data Preparation

To download the dataset, please refer to [prepare_data.md](#).

Self-supervised learning support imagenet(raw and tfrecord) format data.

6.1.1 Imagenet format

You can download Imagenet data or use your own unlabeled image data. You should provide a directory which contains images for self-supervised training and a filelist which contains image path to the root directory. For example, the image directory is as follows

```
images/
├── 0001.jpg
├── 0002.jpg
├── 0003.jpg
├── ...
└── 9999.jpg
```

the content of filelist is

```
0001.jpg
0002.jpg
0003.jpg
...
9999.jpg
```

6.2 Local & PAI-DSW

We use `configs/selfsup/mocov2/mocov2_rn50_8xb32_200e_jpg.py` as an example config in which two config variable should be modified

```
data_train_list = 'filelist.txt'
data_train_root = 'images'
```

6.2.1 Training

Single gpu:

```
python tools/train.py \  
    ${CONFIG_PATH} \  
    --work_dir ${WORK_DIR}
```

Multi gpus:

```
bash tools/dist_train.sh \  
    ${NUM_GPUS} \  
    ${CONFIG_PATH} \  
    --work_dir ${WORK_DIR}
```

- NUM_GPUS: number of gpus
- CONFIG_PATH: the config file path of a selfsup method
- WORK_DIR: your path to save models and logs

Examples:

Edit data_rootpath in the \${CONFIG_PATH} to your own data path.

```
GPUS=8  
bash tools/dist_train.sh configs/selfsup/mocov2/mocov2_rn50_8xb32_200e_jpg.py $GPUS
```

6.2.2 Export model

```
python tools/export.py \  
    ${CONFIG_PATH} \  
    ${CHECKPOINT} \  
    ${EXPORT_PATH}
```

- CONFIG_PATH: the config file path of a selfsup method
- CHECKPOINT: your checkpoint file of a selfsup method named as epoch_*.pth
- EXPORT_PATH: your path to save export model

Examples:

```
python tools/export.py configs/selfsup/mocov2/mocov2_rn50_8xb32_200e_jpg.py \  
    work_dirs/selfsup/mocov2/epoch_200.pth \  
    work_dirs/selfsup/mocov2/epoch_200_export.pth
```

6.2.3 Feature extract

Download `test_image`

```
import cv2
from easycv.predictors.feature_extractor import TorchFeatureExtractor

output_ckpt = 'work_dirs/selfsup/mocov2/epoch_200_export.pth'
fe = TorchFeatureExtractor(output_ckpt)

img = cv2.imread('248347732153_1040.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
feature = fe.predict([img])
print(feature[0]['feature'].shape)
```


YOLOX TUTORIAL

7.1 Data preparation

To download the dataset, please refer to *prepare_data.md*.

Yolox support both coco format and PAI-Itag detection format,

7.1.1 COCO format

To use coco data to train detection, you can refer to `configs/detection/yolox/yolox_s_8xb16_300e_coco.py` for more configuration details.

7.1.2 PAI-Itag detection format

To use pai-itag detection format data to train detection, you can refer to `configs/detection/yolox/yolox_s_8xb16_300e_coco_pai.py` for more configuration details.

7.2 Local & PAI-DSW

To use COCO format data, use config file `configs/detection/yolox/yolox_s_8xb16_300e_coco.py`

To use PAI-Itag format data, use config file `configs/detection/yolox/yolox_s_8xb16_300e_coco_pai.py`

7.2.1 Train

Single gpu:

```
python tools/train.py \
    ${CONFIG_PATH} \
    --work_dir ${WORK_DIR}
```

Multi gpus:

```
bash tools/dist_train.sh \
    ${NUM_GPUS} \
    ${CONFIG_PATH} \
    --work_dir ${WORK_DIR}
```

- NUM_GPUS: number of gpus
- CONFIG_PATH: the config file path of a detection method
- WORK_DIR: your path to save models and logs

Examples:

Edit data_rootpath in the \${CONFIG_PATH} to your own data path.

```
GPUS=8
bash tools/dist_train.sh configs/detection/yolox/yolox_s_8xb16_300e_coco.py $GPUS
```

7.2.2 Evaluation

Single gpu:

```
python tools/eval.py \
    ${CONFIG_PATH} \
    ${CHECKPOINT} \
    --eval
```

Multi gpus:

```
bash tools/dist_test.sh \
    ${CONFIG_PATH} \
    ${NUM_GPUS} \
    ${CHECKPOINT} \
    --eval
```

- CONFIG_PATH: the config file path of a detection method
- NUM_GPUS: number of gpus
- CHECKPOINT: the checkpoint file named as epoch_*.pth.

Examples:

```
GPUS=8
bash tools/dist_test.sh configs/detection/yolox/yolox_s_8xb16_300e_coco.py $GPUS work_
↪ dirs/detection/yolox/epoch_300.pth --eval
```

7.2.3 Export model

```
python tools/export.py \
    ${CONFIG_PATH} \
    ${CHECKPOINT} \
    ${EXPORT_PATH}
```

- CONFIG_PATH: the config file path of a detection method
- CHECKPOINT: your checkpoint file of a detection method named as epoch_*.pth.
- EXPORT_PATH: your path to save export model

Examples:

```
python tools/export.py configs/detection/yolox/yolox_s_8xb16_300e_coco.py \
    work_dirs/detection/yolox/epoch_300.pth \
    work_dirs/detection/yolox/epoch_300_export.pth
```

7.2.4 Inference

Download [test_image](#)

```
import cv2
from easycv.predictors import TorchYoloXPredictor

output_ckpt = 'work_dirs/detection/yolox/epoch_300.pth'
detector = TorchYoloXPredictor(output_ckpt)

img = cv2.imread('0000000017627.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
output = detector.predict([img])
print(output)

# visualize image
from matplotlib import pyplot as plt
image = img.copy()
for box, cls_name in zip(output[0]['detection_boxes'], output[0]['detection_class_names']
    ↳ ' '):
    # box is [x1,y1,x2,y2]
    box = [int(b) for b in box]
    image = cv2.rectangle(image, tuple(box[:2]), tuple(box[2:4]), (0,255,0), 2)
    cv2.putText(image, cls_name, (box[0], box[1]-5), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,
    ↳ 255), 2)
plt.imshow(image)
plt.show()
```


IMAGE CLASSIFICATION TUTORIAL

8.1 Data Preparation

To download the dataset, please refer to [prepare_data.md](#).

Image classification support cifar and imagenet(raw and tfrecord) format data.

8.1.1 Cifar

To use Cifar data to train classification, you can refer to [configs/classification/cifar10/swintiny_b64_5e_jpg.py](#) for more configuration details.

8.1.2 Imagenet format

You can also use your self-defined data which follows `imagenet` format, you should provide a root directory which contains images for classification training and a filelist which contains image path to the root directory. For example, the image root directory is as follows

```
images/
├── 0001.jpg
├── 0002.jpg
├── 0003.jpg
├── ...
└── 9999.jpg
```

each line of the filelist consists of two parts, subpath to the image files starting from the image root directory, class label string for the corresponding image, which are separated by space

```
0001.jpg label1
0002.jpg label2
0003.jpg label3
...
9999.jpg label9999
```

To use Imagenet format data to train classification, you can refer to [configs/classification/imagenet/imagenet_rn50_jpg.py](#) for more configuration details.

8.2 Local & PAI-DSW

8.2.1 Training

Single gpu:

```
python tools/train.py \  
    ${CONFIG_PATH} \  
    --work_dir ${WORK_DIR}
```

Multi gpus:

```
bash tools/dist_train.sh \  
    ${NUM_GPUS} \  
    ${CONFIG_PATH} \  
    --work_dir ${WORK_DIR}
```

- NUM_GPUS: number of gpus
- CONFIG_PATH: the config file path of a image classification method
- WORK_DIR: your path to save models and logs

Examples:

Edit data_rootpath in the \${CONFIG_PATH} to your own data path.

```
single gpu training:  
```shell  
python tools/train.py configs/classification/cifar10/swintiny_b64_5e_jpg.py --work_dir_
↪ work_dirs/classification/cifar10/swintiny --fp16
```  
  
multi gpu training  
```shell  
GPUS=8
bash tools/dist_train.sh configs/classification/cifar10/swintiny_b64_5e_jpg.py $GPUS --
↪ fp16
```  
  
training using python api  
```python  
import easycv.tools

import os
config_path can be a local file or http url
config_path = 'configs/classification/cifar10/swintiny_b64_5e_jpg.py'
easycv.tools.train(config_path, gpus=8, fp16=False, master_port=29527)
```
```

8.2.2 Evaluation

Single gpu:

```
python tools/eval.py \
    ${CONFIG_PATH} \
    ${CHECKPOINT} \
    --eval
```

Multi gpus:

```
bash tools/dist_test.sh \
    ${CONFIG_PATH} \
    ${NUM_GPUS} \
    ${CHECKPOINT} \
    --eval
```

- CONFIG_PATH: the config file path of a image classification method
- NUM_GPUS: number of gpus
- CHECKPOINT: the checkpoint file named as epoch_*.pth

Examples:

```
single gpu evaluation
```shell
python tools/eval.py configs/classification/cifar10/swintiny_b64_5e_jpg.py work_dirs/
→classification/cifar10/swintiny/epoch_350.pth --eval --fp16
```

multi-gpu evaluation

```shell
GPUS=8
bash tools/dist_test.sh configs/classification/cifar10/swintiny_b64_5e_jpg.py $GPUS work_
→dirs/classification/cifar10/swintiny/epoch_350.pth --eval --fp16
```

evaluation using python api
```python
import easycv.tools

import os
os.environ['CUDA_VISIBLE_DEVICES']='3,4,5,6'
config_path = 'configs/classification/cifar10/swintiny_b64_5e_jpg.py'
checkpoint_path = 'work_dirs/classification/cifar10/swintiny/epoch_350.pth'
easycv.tools.eval(config_path, checkpoint_path, gpus=8)
```
```

8.2.3 Export model for inference

```
If SyncBN is configured, we should replace it with BN in config file
```python
imagenet_rn50.py
model = dict(
 ...
 backbone=dict(
 ...
 norm_cfg=dict(type='BN')), # SyncBN --> BN
 ...)
...

```shell
python tools/export.py configs/classification/cifar10/swintiny_b64_5e_jpg.py \
    work_dirs/classification/cifar10/swintiny/epoch_350.pth \
    work_dirs/classification/cifar10/swintiny/epoch_350_export.pth
...

or using python api
```python
import easycv.tools

config_path = './imagenet_rn50.py'
checkpoint_path = 'oss://pai-vision-data-hz/pretrained_models/easycv/resnet/resnet50.pth'
export_path = './resnet50_export.pt'
easycv.tools.export(config_path, checkpoint_path, export_path)
```
```

8.2.4 Inference

Download [test_image](http://pai-vision-data-hz.oss-cn-zhangjiakou.aliyuncs.com/data/cifar10/qince_data/predict/aeroplane_s_000004.png)

```
```python
import cv2
from easycv.predictors.classifier import TorchClassifier

output_ckpt = 'work_dirs/classification/cifar10/swintiny/epoch_350_export.pth'
tcls = TorchClassifier(output_ckpt)

img = cv2.imread('aeroplane_s_000004.png')
input image should be RGB order
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
output = tcls.predict([img])
print(output)
```
```


FILE TUTORIAL

The file module of easycv supports operations both on local and oss files, oss introduction please refer to: <https://www.aliyun.com/product/oss>.

If you operate oss files, you need refer to [access_oss](# access_oss) to authorize oss first.

9.1 Support operations

9.1.1 access_oss

Authorize oss.

Method1:

```
from easycv.file import io
io.access_oss(
    ak_id='your_accesskey_id',
    ak_secret='your_accesskey_secret',
    hosts='your_endpoint' or ['your_endpoint1', 'your_endpoint2'],
    buckets='your_bucket' or ['your_bucket1', 'your_bucket2'])
```

Method2:

Add oss config to your local file ~/.ossutilconfig, as follows: More oss config information, please refer to: https://help.aliyun.com/document_detail/120072.html

```
[Credentials]
language = CH
endpoint = your_endpoint
accessKeyID = your_accesskey_id
accessKeySecret = your_accesskey_secret
[Bucket-Endpoint]
bucket1 = endpoint1
bucket2 = endpoint2
```

Then run the following command, the config file will be read by default to authorize oss.

```
from easycv.file import io
io.access_oss()
```

9.1.2 open

Support w,wb, a, r, rb modes on oss path. Local path is the same usage as the python build-in open.

Example for oss:

io.access_oss please refer to [access_oss](# access_oss).

```
from easycv.file import io

io.access_oss('your oss config')

# Write something to a oss file.
with io.open('oss://bucket_name/demo.txt', 'w') as f:
    f.write("test")

# Read from a oss file.
with io.open('oss://bucket_name/demo.txt', 'r') as f:
    print(f.read())
```

Example for local:

```
from easycv.file import io

# Write something to a oss file.
with io.open('/your/local/path/demo.txt', 'w') as f:
    f.write("test")

# Read from a oss file.
with io.open('/your/local/path/demo.txt', 'r') as f:
    print(f.read())
```

9.1.3 exists

Whether the file exists, same usage as os.path.exists. Support local path and oss path.

Example for oss:

io.access_oss please refer to [access_oss](# access_oss).

```
from easycv.file import io

io.access_oss('your oss config')

ret = io.exists('oss://bucket_name/dir')
print(ret)
```

Example for Local:

```
from easycv.file import io

ret = io.exists('oss://bucket_name/dir')
print(ret)
```

9.1.4 move

Move src to dst, same usage as `shutil.move`. Support local path and oss path.

Example for oss:

`io.access_oss` please refer to `[access_oss](# access_oss)`.

```
from easycv.file import io

io.access_oss('your oss config')

# move oss file to local
io.move('oss://bucket_name/file.txt', '/your/local/path/file.txt')
# move oss file to oss
io.move('oss://bucket_name/dir1/file.txt', 'oss://bucket_name/dir2/file.txt')
# move local file to oss
io.move('/your/local/file.txt', 'oss://bucket_name/file.txt')
# move directory
io.move('oss://bucket_name/dir1/', 'oss://bucket_name/dir2/')
```

Example for local:

```
from easycv.file import io

# move local file to local
io.move('/your/local/path1/file.txt', '/your/local/path2/file.txt')
# move local dir to local
io.move('/your/local/dir1', '/your/local/dir2')
```

9.1.5 copy

Copy a file from src to dst. Same usage as `shutil.copyfile`. If you want to copy a directory, please refer to `[copy-tree](# copytree)`.

Example for oss:

`io.access_oss` please refer to `[access_oss](# access_oss)`.

```
from easycv.file import io

io.access_oss('your oss config')

# Copy a file from local to oss:
io.copy('/your/local/file.txt', 'oss://bucket/dir/file.txt')
# Copy a oss file to local:
io.copy('oss://bucket/dir/file.txt', '/your/local/file.txt')
# Copy a file from oss to oss::
io.copy('oss://bucket/dir/file.txt', 'oss://bucket/dir/file2.txt')
```

Example for local:

```
from easycv.file import io
```

(continues on next page)

(continued from previous page)

```
# Copy a file from local to local:
io.copy('/your/local/path1/file.txt', '/your/local/path2/file.txt')
```

9.1.6 copytree

Copy files recursively from src to dst. Same usage as `shutil.copytree`.

If you want to copy a file, please use `[copy](# copy)`.

Example for oss:

`io.access_oss` please refer to `[access_oss](# access_oss)`.

```
from easycv.file import io

io.access_oss('your oss config')

# copy files from local to oss
io.copytree(src='/your/local/dir1', dst='oss://bucket_name/dir2')
# copy files from oss to local
io.copytree(src='oss://bucket_name/dir2', dst='/your/local/dir1')
# copy files from oss to oss
io.copytree(src='oss://bucket_name/dir1', dst='oss://bucket_name/dir2')
```

Example for local:

```
from easycv.file import io

# copy files from local to local
io.copytree(src='/your/local/dir1', dst='/your/local/dir2')
```

9.1.7 listdir

List all objects in path. Same usage as `os.listdir`.

Example for oss:

`io.access_oss` please refer to `[access_oss](# access_oss)`.

```
from easycv.file import io

io.access_oss('your oss config')

ret = io.listdir('oss://bucket/dir', recursive=True)
print(ret)
```

Example for local:

```
from easycv.file import io

ret = io.listdir('oss://bucket/dir', recursive=True)
print(ret)
```

9.1.8 remove

Remove a file or a directory recursively. Same usage as `os.remove` or `shutil.rmtree`.

Example for oss:

`io.access_oss` please refer to `[access_oss](# access_oss)`.

```
from easycv.file import io

io.access_oss('your oss config')

# Remove a oss file
io.remove('oss://bucket_name/file.txt')
# Remove a oss directory
io.remove('oss://bucket_name/dir/')
```

Example for local:

```
from easycv.file import io

# Remove a local file
io.remove('/your/local/path/file.txt')
# Remove a local directory
io.remove('/your/local/dir/')
```

9.1.9 rmtree

Remove directory recursively, same usage as `shutil.rmtree`.

Example for oss:

`io.access_oss` please refer to `[access_oss](# access_oss)`.

```
from easycv.file import io

io.access_oss('your oss config')

io.remove('oss://bucket_name/dir_name/')
```

Example for local:

```
from easycv.file import io

io.remove('/your/local/dir/')
```

9.1.10 makedirs

Create directories recursively, same usage as `os.makedirs`.

Example for oss:

`io.access_oss` please refer to `[access_oss](# access_oss)`.

```
from easycv.file import io

io.access_oss('your oss config')

io.makedirs('oss://bucket/new_dir/')
```

Example for local:

```
from easycv.file import io

io.makedirs('/your/local/new_dir/')
```

9.1.11 isdir

Return whether a path is directory, same usage as `os.path.isdir`.

Example for oss:

`io.access_oss` please refer to `[access_oss](# access_oss)`.

```
from easycv.file import io

io.access_oss('your oss config') # only oss file need, refer to `IO.access_oss`
ret = io.isdir('oss://bucket/dir/')
print(ret)
```

Example for local:

```
from easycv.file import io

ret = io.isdir('your/local/dir/')
print(ret)
```

9.1.12 isfile

Return whether a path is file object, same usage as `os.path.isfile`.

Example for oss:

`io.access_oss` please refer to `[access_oss](# access_oss)`.

```
from easycv.file import io

io.access_oss('your oss config')
ret = io.isfile('oss://bucket/file.txt')
print(ret)
```

Example for local:

```
from easycv.file import io

ret = io.isfile('/your/local/path/file.txt')
print(ret)
```

9.1.13 glob

Return a list of paths matching a pathname pattern.

Example for oss:

io.access_oss please refer to [access_oss](# access_oss).

```
from easycv.file import io

io.access_oss('your oss config')

ret = io.glob('oss://bucket/dir/*.txt')
print(ret)
```

Example for local:

```
from easycv.file import io

ret = io.glob('/your/local/dir/*.txt')
print(ret)
```

9.1.14 size

Get the size of file path, same usage as `os.path.getsize`.

Example for oss:

io.access_oss please refer to [access_oss](# access_oss).

```
from easycv.file import io

io.access_oss('your oss config')

size = io.size('oss://bucket/file.txt')
print(size)
```

Example for local:

```
from easycv.file import io

size = io.size('/your/local/path/file.txt')
print(size)
```


- initial commit & first release

1. SOTA SSL Algorithms

EasyCV provides state-of-the-art algorithms in self-supervised learning based on contrastive learning such as SimCLR, MoCO V2, Swav, DINO and also MAE based on masked image modeling. We also provides standard benchmark tools for ssl model evaluation.

1. Vision Transformers

EasyCV aims to provide plenty vision transformer models trained either using supervised learning or self-supervised learning, such as ViT, Swin-Transformer and Xcit. More models will be added in the future.

1. Functionality & Extensibility

In addition to SSL, EasyCV also support image classification, object detection, metric learning, and more area will be supported in the future. Although converging different area, EasyCV decompose the framework into different componets such as dataset, model, running hook, making it easy to add new componets and combining it with existing modules. EasyCV provide simple and comprehensive interface for inference. Additionally, all models are supported on PAI-EAS, which can be easily deployed as online service and support automatic scaling and service moniting.

1. Efficiency

EasyCV support multi-gpu and multi worker training. EasyCV use DALI to accelerate data io and preprocessing process, and use fp16 to accelerate training process. For inference optimization, EasyCV export model using jit script, which can be optimized by PAI-Blade.

EASYCV.APIS PACKAGE

11.1 Submodules

11.2 easycv.apis.export module

`easycv.apis.export.export(cfg, ckpt_path, filename)`
export model for inference

Parameters

- **cfg** – Config object
- **ckpt_path** (*str*) – path to checkpoint file
- **filename** (*str*) – filename to save exported models

11.3 easycv.apis.test module

`easycv.apis.test.single_cpu_test(model, data_loader, mode='test', show=False, out_dir=None, show_score_thr=0.3, **kwargs)`

`easycv.apis.test.single_gpu_test(model, data_loader, mode='test', use_fp16=False, **kwargs)`
Test model with single.

This method tests model with single

Parameters

- **model** (*str*) – Model to be tested.
- **data_loader** (*nn.DataLoader*) – Pytorch data loader.
- **model** – mode for model to forward
- **use_fp16** – Use fp16 inference

Returns The prediction results.

Return type list

`easycv.apis.test.multi_gpu_test(model, data_loader, mode='test', tmpdir=None, gpu_collect=False, use_fp16=False, **kwargs)`

Test model with multiple gpus.

This method tests model with multiple gpus and collects the results under two different modes: gpu and cpu modes. By setting 'gpu_collect=True' it encodes results to gpu tensors and use gpu communication for results collection. On cpu mode it saves the results on different gpus to 'tmpdir' and collects them by the rank 0 worker.

Parameters

- **model** (*str*) – Model to be tested.
- **data_loader** (*nn.DataLoader*) – Pytorch data loader.
- **model** – mode for model to forward
- **tmpdir** (*str*) – Path of directory to save the temporary results from different gpus under cpu mode.
- **gpu_collect** (*bool*) – Option to use either gpu or cpu to collect results.
- **use_fp16** – Use fp16 inference

Returns The prediction results.

Return type list

```
easycv.apis.test.collect_results_cpu(result_part, size, tmpdir=None)
```

```
easycv.apis.test.serialize_tensor(tensor_collection)
```

```
easycv.apis.test.collect_results_gpu(result_part, size)
```

11.4 easycv.apis.train module

```
easycv.apis.train.set_random_seed(seed, deterministic=False)
```

Set random seed.

Parameters

- **seed** (*int*) – Seed to be used.
- **deterministic** (*bool*) – Whether to set the deterministic option for CUDNN backend, i.e., set *torch.backends.cudnn.deterministic* to True and *torch.backends.cudnn.benchmark* to False. Default: False.

```
easycv.apis.train.train_model(model, data_loaders, cfg, distributed=False, timestamp=None, meta=None, use_fp16=False, validate=True, gpu_collect=True)
```

Training API.

Parameters

- **model** (*nn.Module*) – user defined model
- **data_loaders** – a list of dataloader for training data
- **cfg** – config object
- **distributed** – distributed training or not
- **timestamp** – time str formatted as '%Y%m%d_%H%M%S'
- **meta** – a dict containing meta data info, such as env_info, seed, iter, epoch
- **use_fp16** – use fp16 training or not
- **validate** – do evaluation while training
- **gpu_collect** – use gpu collect or cpu collect for tensor gathering

`easycv.apis.train.get_skip_list_keywords(model)`

`easycv.apis.train.build_optimizer(model, optimizer_cfg)`

Build optimizer from configs.

Parameters

- **model** (`nn.Module`) – The model with parameters to be optimized.
- **optimizer_cfg** (`dict`) – The config dict of the optimizer.

Positional fields are:

- `type`: class name of the optimizer.
- `lr`: base learning rate.

Optional fields are:

- any arguments of the corresponding optimizer type, e.g., `weight_decay`, `momentum`, etc.
- `paramwise_options`: a dict with regular expression as keys to match parameter names and a dict containing options as values. Options include 6 fields: `lr`, `lr_mult`, `momentum`, `momentum_mult`, `weight_decay`, `weight_decay_mult`.

Returns The initialized optimizer.

Return type `torch.optim.Optimizer`

Example

```
>>> model = torch.nn.modules.Conv1d(1, 1, 1)
>>> paramwise_options = {
>>>     '(bn|gn)(\d+)?.(weight|bias)': dict(weight_decay_mult=0.1),
>>>     '\Ahead.': dict(lr_mult=10, momentum=0)}
>>> optimizer_cfg = dict(type='SGD', lr=0.01, momentum=0.9,
>>>                        weight_decay=0.0001,
>>>                        paramwise_options=paramwise_options)
>>> optimizer = build_optimizer(model, optimizer_cfg)
```

11.5 easycv.apis.train_misc module

`easycv.apis.train_misc.build_yolo_optimizer(model, optimizer_cfg)`

build optimizer for yolo.

EASYCV.DATASETS PACKAGE

12.1 Subpackages

12.1.1 easycv.datasets.classification package

class `easycv.datasets.classification.ClsDataset(data_source, pipeline)`

Bases: `Generic[torch.utils.data.dataset.T_co]`

Dataset for classification

Parameters

- **data_source** – data source to parse input data
- **pipeline** – transforms list

__init__(*data_source*, *pipeline*)

Initialize self. See `help(type(self))` for accurate signature.

evaluate(*results*, *evaluators*, *logger=None*, *topk=(1, 5)*)

evaluate classification task

Parameters

- **results** – a dict of list of tensor, including prediction and groundtruth info, where prediction tensor is `NxC` and the same with groundtruth labels.
- **evaluators** – a list of evaluator

Returns a dict of float, different metric values

Return type `eval_result`

visualize(*results*, *vis_num=10*, ***kwargs*)

Visualize the model output on validation data. :param results: A dictionary containing

class: List of length number of test images. `img metas`: List of length number of test images, dict of image meta info, containing filename, `img_shape`, `origin_img_shape` and so on.

Parameters **vis_num** – number of images visualized

Returns: A dictionary containing `images`: Visualized images, list of `np.ndarray`. `img metas`: List of length number of test images,

dict of image meta info, containing filename, `img_shape`, `origin_img_shape` and so on.

```
class easycv.datasets.classification.ClsOddsDataset(data_source, pipeline, image_key='url_image',  
                                                  label_key='label', **kwargs)  
  
    Bases: Generic[torch.utils.data.dataset.T_co]  
  
    Dataset for rotation prediction  
  
    __init__(data_source, pipeline, image_key='url_image', label_key='label', **kwargs)  
        Initialize self. See help(type(self)) for accurate signature.  
  
    evaluate(results, evaluators, logger=None)
```

Subpackages

easycv.datasets.classification.data_sources package

```
class easycv.datasets.classification.data_sources.ClsSourceCifar10(root, split)  
    Bases: object  
  
    CLASSES = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse',  
               'ship', 'truck']  
  
    __init__(root, split)  
        Initialize self. See help(type(self)) for accurate signature.  
  
    get_length()  
  
    get_sample(idx)
```

```
class easycv.datasets.classification.data_sources.ClsSourceCifar100(root, split)  
    Bases: object  
  
    CLASSES = None  
  
    __init__(root, split)  
        Initialize self. See help(type(self)) for accurate signature.  
  
    get_length()  
  
    get_sample(idx)
```

```
class easycv.datasets.classification.data_sources.ClsSourceImageListByClass(root, list_file,  
                                                                           m_per_class=2,  
                                                                           delimiter=' ',  
                                                                           split_huge_listfile_byrank=False,  
                                                                           cache_path='data/',  
                                                                           max_try=20)
```

Bases: object

Get the same *m_per_class* samples by the label idx.

Parameters

- **list_file** – str / list(str), str means a input image list file path, this file contains records as *image_path label* in list_file list(str) means multi image list, each one contains some records as *image_path label*
- **root** – str / list(str), root path for image_path, each list_file will need a root.
- **m_per_class** – num of samples for each class.
- **delimiter** – str, delimiter of each line in the *list_file*

- **split_huge_listfile_byrank** – Adapt to the situation that the memory cannot fully load a huge amount of data list. If split, data list will be split to each rank.
- **cache_path** – if *split_huge_listfile_byrank* is true, cache list_file will be saved to cache_path.
- **max_try** – int, max try numbers of reading image

```
__init__(root, list_file, m_per_class=2, delimiter=' ', split_huge_listfile_byrank=False, cache_path='data',
          max_try=20)
```

Initialize self. See help(type(self)) for accurate signature.

```
get_length()
```

```
get_sample(idx)
```

```
class easycv.datasets.classification.data_sources.ClsSourceImageList(list_file, root="",
                                                                    delimiter=' ',
                                                                    split_huge_listfile_byrank=False,
                                                                    split_label_balance=False,
                                                                    cache_path='data',
                                                                    max_try=20)
```

Bases: object

data source for classification

Parameters

- **list_file** – str / list(str), str means a input image list file path, this file contains records as *image_path label* in list_file list(str) means multi image list, each one contains some records as *image_path label*
- **root** – str / list(str), root path for image_path, each list_file will need a root, if len(root) < len(list_file), we will use root[-1] to fill root list.
- **delimiter** – str, delimiter of each line in the *list_file*
- **split_huge_listfile_byrank** – Adapt to the situation that the memory cannot fully load a huge amount of data list. If split, data list will be split to each rank.
- **split_label_balance** – if *split_huge_listfile_byrank* is true, whether split with label balance
- **cache_path** – if *split_huge_listfile_byrank* is true, cache list_file will be saved to cache_path.
- **max_try** – int, max try numbers of reading image

```
__init__(list_file, root="", delimiter=' ', split_huge_listfile_byrank=False, split_label_balance=False,
          cache_path='data', max_try=20)
```

Initialize self. See help(type(self)) for accurate signature.

```
static parse_list_file(list_file, root, delimiter)
```

```
get_length()
```

```
get_sample(idx)
```

```
class easycv.datasets.classification.data_sources.ClsSourceImageNetTFRecord(list_file="",
                                                                              root="",
                                                                              file_pattern=None,
                                                                              cache_path='data/cache',
                                                                              max_try=10)
```

Bases: object

data source for imagenet tfrecord.

```
__init__(list_file="", root="", file_pattern=None, cache_path='data/cache/', max_try=10)  
    Initialize self. See help(type(self)) for accurate signature.
```

```
class easycv.datasets.classification.data_sources.ClsSourceCUB(*args, ann_file,  
                                                             image_class_labels_file,  
                                                             train_test_split_file, test_mode,  
                                                             data_prefix, **kwargs)
```

Bases: object

The CUB-200-2011 Dataset. Support the [CUB-200-2011](#) Dataset. Comparing with the [CUB-200](#) Dataset, there are much more pictures in *CUB-200-2011*. :param ann_file: the annotation file.

images.txt in CUB.

Parameters

- **image_class_labels_file** (*str*) – the label file. image_class_labels.txt in CUB.
- **train_test_split_file** (*str*) – the split file. train_test_split_file.txt in CUB.

```

CLASSES = ['Black_footed_Albatross', 'Laysan_Albatross', 'Sooty_Albatross',
'Groove_billed_Ani', 'Crested_Auklet', 'Least_Auklet', 'Parakeet_Auklet',
'Rhinoceros_Auklet', 'Brewer_Blackbird', 'Red_winged_Blackbird', 'Rusty_Blackbird',
'Yellow_headed_Blackbird', 'Bobolink', 'Indigo_Bunting', 'Lazuli_Bunting',
'Painted_Bunting', 'Cardinal', 'Spotted_Catbird', 'Gray_Catbird',
'Yellow_breasted_Chat', 'Eastern_Towhee', 'Chuck_will_Widow', 'Brandt_Cormorant',
'Red_faced_Cormorant', 'Pelagic_Cormorant', 'Bronzed_Cowbird', 'Shiny_Cowbird',
'Brown_Creeper', 'American_Crow', 'Fish_Crow', 'Black_billed_Cuckoo',
'Mangrove_Cuckoo', 'Yellow_billed_Cuckoo', 'Gray_crowned_Rosy_Finch',
'Purple_Finch', 'Northern_Flicker', 'Acadian_Flycatcher',
'Great_Crested_Flycatcher', 'Least_Flycatcher', 'Olive_sided_Flycatcher',
'Scissor_tailed_Flycatcher', 'Vermilion_Flycatcher', 'Yellow_bellied_Flycatcher',
'Frigatebird', 'Northern_Fulmar', 'Gadwall', 'American_Goldfinch',
'European_Goldfinch', 'Boat_tailed_Grackle', 'Eared_Grebe', 'Horned_Grebe',
'Pied_billed_Grebe', 'Western_Grebe', 'Blue_Grosbeak', 'Evening_Grosbeak',
'Pine_Grosbeak', 'Rose_breasted_Grosbeak', 'Pigeon_Guillemot', 'California_Gull',
'Glaucous_winged_Gull', 'Heermann_Gull', 'Herring_Gull', 'Ivory_Gull',
'Ring_billed_Gull', 'Slaty_backed_Gull', 'Western_Gull', 'Anna_Hummingbird',
'Ruby_throated_Hummingbird', 'Rufous_Hummingbird', 'Green_Violetear',
'Long_tailed_Jaeger', 'Pomarine_Jaeger', 'Blue_Jay', 'Florida_Jay', 'Green_Jay',
'Dark_eyed_Junco', 'Tropical_Kingbird', 'Gray_Kingbird', 'Belted_Kingfisher',
'Green_Kingfisher', 'Pied_Kingfisher', 'Ringed_Kingfisher',
'White_breasted_Kingfisher', 'Red_legged_Kittiwake', 'Horned_Lark', 'Pacific_Loon',
'Mallard', 'Western_Meadowlark', 'Hooded_Merganser', 'Red_breasted_Merganser',
'Mockingbird', 'Nighthawk', 'Clark_Nutcracker', 'White_breasted_Nuthatch',
'Baltimore_Oriole', 'Hooded_Oriole', 'Orchard_Oriole', 'Scott_Oriole', 'Ovenbird',
'Brown_Pelican', 'White_Pelican', 'Western_Wood_Pewee', 'Sayornis',
'American_Pipit', 'Whip_poor_Will', 'Horned_Puffin', 'Common_Raven',
'White_necked_Raven', 'American_Redstart', 'Geococcyx', 'Loggerhead_Shrike',
'Great_Grey_Shrike', 'Baird_Sparrow', 'Black_throated_Sparrow', 'Brewer_Sparrow',
'Chipping_Sparrow', 'Clay_colored_Sparrow', 'House_Sparrow', 'Field_Sparrow',
'Fox_Sparrow', 'Grasshopper_Sparrow', 'Harris_Sparrow', 'Henslow_Sparrow',
'Le_Conte_Sparrow', 'Lincoln_Sparrow', 'Nelson_Sharp_tailed_Sparrow',
'Savannah_Sparrow', 'Seaside_Sparrow', 'Song_Sparrow', 'Tree_Sparrow',
'Vesper_Sparrow', 'White_crowned_Sparrow', 'White_throated_Sparrow',
'Cape_Glossy_Starling', 'Bank_Swallow', 'Barn_Swallow', 'Cliff_Swallow',
'Tree_Swallow', 'Scarlet_Tanager', 'Summer_Tanager', 'Arctic_Tern', 'Black_Tern',
'Caspian_Tern', 'Common_Tern', 'Elegant_Tern', 'Forsters_Tern', 'Least_Tern',
'Green_tailed_Towhee', 'Brown_Thrasher', 'Sage_Thrasher', 'Black_capped_Vireo',
'Blue_headed_Vireo', 'Philadelphia_Vireo', 'Red_eyed_Vireo', 'Warbling_Vireo',
'White_eyed_Vireo', 'Yellow_throated_Vireo', 'Bay_breasted_Warbler',
'Black_and_white_Warbler', 'Black_throated_Blue_Warbler', 'Blue_winged_Warbler',
'Canada_Warbler', 'Cape_May_Warbler', 'Cerulean_Warbler', 'Chestnut_sided_Warbler',
'Golden_winged_Warbler', 'Hooded_Warbler', 'Kentucky_Warbler', 'Magnolia_Warbler',
'Mourning_Warbler', 'Myrtle_Warbler', 'Nashville_Warbler', 'Orange_crowned_Warbler',
'Palm_Warbler', 'Pine_Warbler', 'Prairie_Warbler', 'Prothonotary_Warbler',
'Swainson_Warbler', 'Tennessee_Warbler', 'Wilson_Warbler', 'Worm_eating_Warbler',
'Yellow_Warbler', 'Northern_Waterthrush', 'Louisiana_Waterthrush',
'Bohemian_Waxwing', 'Cedar_Waxwing', 'American_Three_toed_Woodpecker',
'Pileated_Woodpecker', 'Red_bellied_Woodpecker', 'Red_cockaded_Woodpecker',
'Red_headed_Woodpecker', 'Downy_Woodpecker', 'Bewick_Wren', 'Cactus_Wren',
'Carolina_Wren', 'House_Wren', 'Marsh_Wren', 'Rock_Wren', 'Winter_Wren',
'Common_Yellowthroat']

```

```
__init__(*args, ann_file, image_class_labels_file, train_test_split_file, test_mode, data_prefix, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.

load_annotations()

get_length()

get_sample(idx)
```

Submodules

easycv.datasets.classification.data_sources.cifar module

```
class easycv.datasets.classification.data_sources.cifar.ClsSourceCifar10(root, split)
    Bases: object

    CLASSES = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse',
               'ship', 'truck']

    __init__(root, split)
        Initialize self. See help(type(self)) for accurate signature.

    get_length()

    get_sample(idx)

class easycv.datasets.classification.data_sources.cifar.ClsSourceCifar100(root, split)
    Bases: object

    CLASSES = None

    __init__(root, split)
        Initialize self. See help(type(self)) for accurate signature.

    get_length()

    get_sample(idx)
```

easycv.datasets.classification.data_sources.class_list module

```
class easycv.datasets.classification.data_sources.class_list.ClsSourceImageListByClass(root,
                                                                                       list_file,
                                                                                       m_per_class=2,
                                                                                       de-
                                                                                       lime-
                                                                                       ter='
                                                                                       ,
                                                                                       split_huge_listfile_by
                                                                                       cache_path='data',
                                                                                       max_try=20)
```

Bases: object

Get the same *m_per_class* samples by the label idx.

Parameters

- **list_file** – str / list(str), str means a input image list file path, this file contains records as *image_path label* in list_file list(str) means multi image list, each one contains some records as *image_path label*

- **root** – str / list(str), root path for image_path, each list_file will need a root.
- **m_per_class** – num of samples for each class.
- **delimiter** – str, delimiter of each line in the *list_file*
- **split_huge_listfile_byrank** – Adapt to the situation that the memory cannot fully load a huge amount of data list. If split, data list will be split to each rank.
- **cache_path** – if *split_huge_listfile_byrank* is true, cache list_file will be saved to cache_path.
- **max_try** – int, max try numbers of reading image

```
__init__(root, list_file, m_per_class=2, delimiter=' ', split_huge_listfile_byrank=False, cache_path='data/',
         max_try=20)
```

Initialize self. See help(type(self)) for accurate signature.

```
get_length()
```

```
get_sample(idx)
```

easycv.datasets.classification.data_sources.fashiongen_h5 module

```
class easycv.datasets.classification.data_sources.fashiongen_h5.FashionGenH5(h5file_path, re-
                                                                           turn_label=True,
                                                                           cache_path='data/fashionGenH5')
```

Bases: object

```
__init__(h5file_path, return_label=True, cache_path='data/fashionGenH5')
```

Initialize self. See help(type(self)) for accurate signature.

```
get_length()
```

```
get_sample(idx)
```

easycv.datasets.classification.data_sources.image_list module

```
class easycv.datasets.classification.data_sources.image_list.ClsSourceImageList(list_file,
                                                                           root="",
                                                                           delimiter=' ',
                                                                           split_huge_listfile_byrank=False,
                                                                           split_label_balance=False,
                                                                           cache_path='data/',
                                                                           max_try=20)
```

Bases: object

data source for classification

Parameters

- **list_file** – str / list(str), str means a input image list file path, this file contains records as *image_path label* in list_file list(str) means multi image list, each one contains some records as *image_path label*
- **root** – str / list(str), root path for image_path, each list_file will need a root, if len(root) < len(list_file), we will use root[-1] to fill root list.
- **delimiter** – str, delimiter of each line in the *list_file*

- **split_huge_listfile_byrank** – Adapt to the situation that the memory cannot fully load a huge amount of data list. If split, data list will be split to each rank.
- **split_label_balance** – if *split_huge_listfile_byrank* is true, whether split with label balance
- **cache_path** – if *split_huge_listfile_byrank* is true, cache list_file will be saved to cache_path.
- **max_try** – int, max try numbers of reading image

```
__init__(list_file, root="", delimiter=',', split_huge_listfile_byrank=False, split_label_balance=False,
         cache_path='data/', max_try=20)
```

Initialize self. See help(type(self)) for accurate signature.

```
static parse_list_file(list_file, root, delimiter)
```

```
get_length()
```

```
get_sample(idx)
```

easycv.datasets.classification.data_sources.imagenet_tfrecord module

```
class easycv.datasets.classification.data_sources.imagenet_tfrecord.ClsSourceImageNetTFRecord(list_file="",
                                                                                             root="",
                                                                                             file_pattern=
                                                                                             cache_path=
                                                                                             max_try=10)
```

Bases: object

data source for imagenet tfrecord.

```
__init__(list_file="", root="", file_pattern=None, cache_path='data/cache/', max_try=10)
Initialize self. See help(type(self)) for accurate signature.
```

easycv.datasets.classification.data_sources.utils module

```
easycv.datasets.classification.data_sources.utils.split_listfile_byrank(list_file,
                                                                           label_balance,
                                                                           save_path='data/',
                                                                           delimiter=',')
```


easycv.datasets.classification.pipelines package

```

class easycv.datasets.classification.pipelines.MMAutoAugment(policies=[['type': 'Posterize', 'bits':
    4, 'prob': 0.4], ['type': 'Rotate',
    'angle': 30.0, 'prob': 0.6]], [['type':
    'Solarize', 'thr':
    113.77777777777777, 'prob': 0.6],
    ['type': 'AutoContrast', 'prob': 0.6]],
    [['type': 'Equalize', 'prob': 0.8],
    ['type': 'Equalize', 'prob': 0.6]],
    [['type': 'Posterize', 'bits': 5, 'prob':
    0.6], ['type': 'Posterize', 'bits': 5,
    'prob': 0.6]], [['type': 'Equalize',
    'prob': 0.4], ['type': 'Solarize', 'thr':
    142.22222222222223, 'prob': 0.2]],
    [['type': 'Equalize', 'prob': 0.4],
    ['type': 'Rotate', 'angle':
    26.666666666666668, 'prob': 0.8]],
    [['type': 'Solarize', 'thr':
    170.66666666666666, 'prob': 0.6],
    ['type': 'Equalize', 'prob': 0.6]],
    [['type': 'Posterize', 'bits': 6, 'prob':
    0.8], ['type': 'Equalize', 'prob': 1.0]],
    [['type': 'Rotate', 'angle': 10.0,
    'prob': 0.2], ['type': 'Solarize', 'thr':
    28.444444444444443, 'prob': 0.6]],
    [['type': 'Equalize', 'prob': 0.6],
    ['type': 'Posterize', 'bits': 5, 'prob':
    0.4]], [['type': 'Rotate', 'angle':
    26.666666666666668, 'prob': 0.8],
    ['type': 'ColorTransform',
    'magnitude': 0.0, 'prob': 0.4]],
    [['type': 'Rotate', 'angle': 30.0,
    'prob': 0.4], ['type': 'Equalize',
    'prob': 0.6]], [['type': 'Equalize',
    'prob': 0.0], ['type': 'Equalize',
    'prob': 0.8]], [['type': 'Invert', 'prob':
    0.6], ['type': 'Equalize', 'prob': 1.0]],
    [['type': 'ColorTransform',
    'magnitude': 0.4, 'prob': 0.6],
    ['type': 'Contrast', 'magnitude': 0.8,
    'prob': 1.0]], [['type': 'Rotate',
    'angle': 26.666666666666668,
    'prob': 0.8], ['type':
    'ColorTransform', 'magnitude': 0.2,
    'prob': 1.0]], [['type':
    'ColorTransform', 'magnitude': 0.8,
    'prob': 0.8], ['type': 'Solarize', 'thr':
    56.888888888888886, 'prob': 0.8]],
    [['type': 'Sharpness', 'magnitude':
    0.7, 'prob': 0.4], ['type': 'Invert',
    'prob': 0.6]], [['type': 'Shear',
    'magnitude': 0.16666666666666666,
    'prob': 0.6, 'direction': 'horizontal'],
    ['type': 'Equalize', 'prob': 1.0]],
    [['type': 'ColorTransform',
    'magnitude': 0.0, 'prob': 0.4],
    ['type': 'Equalize', 'prob': 0.6]],
    [['type': 'Equalize', 'prob': 0.4],
    ['type': 'Solarize', 'thr':

```


Auto augmentation. This data augmentation is proposed in [AutoAugment: Learning Augmentation Policies from Data](#). :param policies: The policies of auto augmentation. Each

policy in `policies` is a specific augmentation policy, and is composed by several augmentations (dict). When AutoAugment is called, a random policy in `policies` will be selected to augment images.

Parameters `hparams` (*dict*) – Configs of hyperparameters. Hyperparameters will be used in policies that require these arguments if these arguments are not set in policy dicts. Defaults to use `_HPARAMS_DEFAULT`.

```
__init__(policies=[[{'type': 'Posterize', 'bits': 4, 'prob': 0.4}, {'type': 'Rotate', 'angle': 30.0, 'prob': 0.6}],
[{'type': 'Solarize', 'thr': 113.77777777777777, 'prob': 0.6}, {'type': 'AutoContrast', 'prob': 0.6}],
[{'type': 'Equalize', 'prob': 0.8}, {'type': 'Equalize', 'prob': 0.6}], [{'type': 'Posterize', 'bits': 5,
'prob': 0.6}, {'type': 'Posterize', 'bits': 5, 'prob': 0.6}], [{'type': 'Equalize', 'prob': 0.4}, {'type':
'Solarize', 'thr': 142.22222222222223, 'prob': 0.2}], [{'type': 'Equalize', 'prob': 0.4}, {'type':
'Rotate', 'angle': 26.666666666666668, 'prob': 0.8}], [{'type': 'Solarize', 'thr':
170.66666666666666, 'prob': 0.6}, {'type': 'Equalize', 'prob': 0.6}], [{'type': 'Posterize', 'bits': 6,
'prob': 0.8}, {'type': 'Equalize', 'prob': 1.0}], [{'type': 'Rotate', 'angle': 10.0, 'prob': 0.2}, {'type':
'Solarize', 'thr': 28.444444444444443, 'prob': 0.6}], [{'type': 'Equalize', 'prob': 0.6}, {'type':
'Posterize', 'bits': 5, 'prob': 0.4}], [{'type': 'Rotate', 'angle': 26.666666666666668, 'prob': 0.8},
{'type': 'ColorTransform', 'magnitude': 0.0, 'prob': 0.4}], [{'type': 'Rotate', 'angle': 30.0, 'prob':
0.4}, {'type': 'Equalize', 'prob': 0.6}], [{'type': 'Equalize', 'prob': 0.0}, {'type': 'Equalize', 'prob':
0.8}], [{'type': 'Invert', 'prob': 0.6}, {'type': 'Equalize', 'prob': 1.0}], [{'type': 'ColorTransform',
'magnitude': 0.4, 'prob': 0.6}, {'type': 'Contrast', 'magnitude': 0.8, 'prob': 1.0}], [{'type': 'Rotate',
'angle': 26.666666666666668, 'prob': 0.8}, {'type': 'ColorTransform', 'magnitude': 0.2, 'prob':
1.0}], [{'type': 'ColorTransform', 'magnitude': 0.8, 'prob': 0.8}, {'type': 'Solarize', 'thr':
56.888888888888886, 'prob': 0.8}], [{'type': 'Sharpness', 'magnitude': 0.7, 'prob': 0.4}, {'type':
'Invert', 'prob': 0.6}], [{'type': 'Shear', 'magnitude': 0.16666666666666666, 'prob': 0.6, 'direction':
'horizontal'}, {'type': 'Equalize', 'prob': 1.0}], [{'type': 'ColorTransform', 'magnitude': 0.0, 'prob':
0.4}, {'type': 'Equalize', 'prob': 0.6}], [{'type': 'Equalize', 'prob': 0.4}, {'type': 'Solarize', 'thr':
142.22222222222223, 'prob': 0.2}], [{'type': 'Solarize', 'thr': 113.77777777777777, 'prob': 0.6},
{'type': 'AutoContrast', 'prob': 0.6}], [{'type': 'Invert', 'prob': 0.6}, {'type': 'Equalize', 'prob': 1.0}],
[{'type': 'ColorTransform', 'magnitude': 0.4, 'prob': 0.6}, {'type': 'Contrast', 'magnitude': 0.8,
'prob': 1.0}], [{'type': 'Equalize', 'prob': 0.8}, {'type': 'Equalize', 'prob': 0.6}]],
hparams={'pad_val': 128})
```

Initialize self. See `help(type(self))` for accurate signature.

```
class easycv.datasets.classification.pipelines.MMRandAugment(num_policies, magnitude_level,
                                                            magnitude_std=0.0, total_level=30,
                                                            policies=[{'type': 'AutoContrast'},
                                                            {'type': 'Equalize'}, {'type': 'Invert'},
                                                            {'type': 'Rotate', 'magnitude_key':
                                                            'angle', 'magnitude_range': (0, 30)},
                                                            {'type': 'Posterize', 'magnitude_key':
                                                            'bits', 'magnitude_range': (4, 0)},
                                                            {'type': 'Solarize', 'magnitude_key':
                                                            'thr', 'magnitude_range': (256, 0)},
                                                            {'type': 'SolarizeAdd',
                                                            'magnitude_key': 'magnitude',
                                                            'magnitude_range': (0, 110)}, {'type':
                                                            'ColorTransform', 'magnitude_key':
                                                            'magnitude', 'magnitude_range': (0,
                                                            0.9)}, {'type': 'Contrast',
                                                            'magnitude_key': 'magnitude',
                                                            'magnitude_range': (0, 0.9)}, {'type':
                                                            'Brightness', 'magnitude_key':
                                                            'magnitude', 'magnitude_range': (0,
                                                            0.9)}, {'type': 'Sharpness',
                                                            'magnitude_key': 'magnitude',
                                                            'magnitude_range': (0, 0.9)}, {'type':
                                                            'Shear', 'magnitude_key':
                                                            'magnitude', 'magnitude_range': (0,
                                                            0.3), 'direction': 'horizontal'},
                                                            {'type': 'Shear', 'magnitude_key':
                                                            'magnitude', 'magnitude_range': (0,
                                                            0.3), 'direction': 'vertical'}, {'type':
                                                            'Translate', 'magnitude_key':
                                                            'magnitude', 'magnitude_range': (0,
                                                            0.45), 'direction': 'horizontal'},
                                                            {'type': 'Translate', 'magnitude_key':
                                                            'magnitude', 'magnitude_range': (0,
                                                            0.45), 'direction': 'vertical'}],
                                                            hparams={'pad_val': 128})
```

Bases: object

Random augmentation. This data augmentation is proposed in [RandAugment: Practical automated data augmentation with a reduced search space](#). :param policies: The policies of random augmentation. Each

policy in `policies` is one specific augmentation policy (dict). The policy shall at least have key `type`, indicating the type of augmentation. For those which have magnitude, (given to the fact they are named differently in different augmentation,) `magnitude_key` and `magnitude_range` shall be the magnitude argument (str) and the range of magnitude (tuple in the format of (val1, val2)), respectively. Note that val1 is not necessarily less than val2.

Parameters

- **num_policies** (*int*) – Number of policies to select from policies each time.
- **magnitude_level** (*int* | *float*) – Magnitude level for all the augmentation selected.
- **total_level** (*int* | *float*) – Total level for the magnitude. Defaults to 30.
- **magnitude_std** (*Number* | *str*) – Deviation of magnitude noise applied. - If positive number, magnitude is sampled from normal distribution

(mean=magnitude, std=magnitude_std).

- If 0 or negative number, magnitude remains unchanged.
- If str “inf”, magnitude is sampled from uniform distribution (range=[min, magnitude]).
- **hparams** (*dict*) – Configs of hyperparameters. Hyperparameters will be used in policies that require these arguments if these arguments are not set in policy dicts. Defaults to use `_HPARAMS_DEFAULT`.

Note: *magnitude_std* will introduce some randomness to policy, modified by <https://github.com/rwightman/pytorch-image-models>. When *magnitude_std*=0, we calculate the magnitude as follows: .. math:

$$\text{magnitude} = \frac{\text{magnitude_level}}{\text{total_level}} \times (\text{val2} - \text{val1}) + \text{val1}$$

```
__init__(num_policies, magnitude_level, magnitude_std=0.0, total_level=30, policies=[{'type':
'AutoContrast'}, {'type': 'Equalize'}, {'type': 'Invert'}, {'type': 'Rotate', 'magnitude_key': 'angle',
'magnitude_range': (0, 30)}, {'type': 'Posterize', 'magnitude_key': 'bits', 'magnitude_range': (4, 0)},
{'type': 'Solarize', 'magnitude_key': 'thr', 'magnitude_range': (256, 0)}, {'type': 'SolarizeAdd',
'magnitude_key': 'magnitude', 'magnitude_range': (0, 110)}, {'type': 'ColorTransform',
'magnitude_key': 'magnitude', 'magnitude_range': (0, 0.9)}, {'type': 'Contrast', 'magnitude_key':
'magnitude', 'magnitude_range': (0, 0.9)}, {'type': 'Brightness', 'magnitude_key': 'magnitude',
'magnitude_range': (0, 0.9)}, {'type': 'Sharpness', 'magnitude_key': 'magnitude',
'magnitude_range': (0, 0.9)}, {'type': 'Shear', 'magnitude_key': 'magnitude', 'magnitude_range': (0,
0.3), 'direction': 'horizontal'}, {'type': 'Shear', 'magnitude_key': 'magnitude', 'magnitude_range':
(0, 0.3), 'direction': 'vertical'}, {'type': 'Translate', 'magnitude_key': 'magnitude',
'magnitude_range': (0, 0.45), 'direction': 'horizontal'}, {'type': 'Translate', 'magnitude_key':
'magnitude', 'magnitude_range': (0, 0.45), 'direction': 'vertical'}], hparams={'pad_val': 128})
```

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.classification.pipelines.MMRandomErasing(erase_prob=0.5,
min_area_ratio=0.02,
max_area_ratio=0.4,
aspect_range=(0.3,
3.3333333333333335),
mode='const', fill_color=(128,
128, 128), fill_std=None)
```

Bases: object

Randomly selects a rectangle region in an image and erase pixels. :param erase_prob: Probability that image will be randomly erased.

Default: 0.5

Parameters

- **min_area_ratio** (*float*) – Minimum erased area / input image area Default: 0.02
- **max_area_ratio** (*float*) – Maximum erased area / input image area Default: 0.4
- **aspect_range** (*sequence | float*) – Aspect ratio range of erased area. if float, it will be converted to (aspect_ratio, 1/aspect_ratio) Default: (3/10, 10/3)

- **mode** (*str*) – Fill method in erased area, can be: - const (default): All pixels are assign with the same value. - rand: each pixel is assigned with a random value in [0, 255]
- **fill_color** (*sequence / Number*) – Base color filled in erased area. Defaults to (128, 128, 128).
- **fill_std** (*sequence / Number, optional*) – If set and mode is 'rand', fill erased area with random color from normal distribution (mean=fill_color, std=fill_std); If not set, fill erased area with random color from uniform distribution (0~255). Defaults to None.

Note: See [Random Erasing Data Augmentation](#) This paper provided 4 modes: RE-R, RE-M, RE-0, RE-255, and use RE-M as default. The config of these 4 modes are: - RE-R: RandomErasing(mode='rand') - RE-M: RandomErasing(mode='const', fill_color=(123.67, 116.3, 103.5)) - RE-0: RandomErasing(mode='const', fill_color=0) - RE-255: RandomErasing(mode='const', fill_color=255)

__init__ (*erase_prob=0.5, min_area_ratio=0.02, max_area_ratio=0.4, aspect_range=(0.3, 3.3333333333333335), mode='const', fill_color=(128, 128, 128), fill_std=None*)
Initialize self. See help(type(self)) for accurate signature.

Submodules

easycv.datasets.classification.pipelines.auto_augment module

easycv.datasets.classification.pipelines.auto_augment.**random_negative** (*value, random_negative_prob*)

Randomly negate value based on random_negative_prob.

easycv.datasets.classification.pipelines.auto_augment.**merge_hparams** (*policy: dict, hparams: dict*)
Merge hyperparameters into policy config. Only merge partial hyperparameters required of the policy. :param policy: Original policy config dict. :type policy: dict :param hparams: Hyperparameters need to be merged. :type hparams: dict

Returns Policy config dict after adding hparams.

Return type dict

12.1. Subpackages

Auto augmentation. This data augmentation is proposed in [AutoAugment: Learning Augmentation Policies from Data](#). :param policies: The policies of auto augmentation. Each

policy in `policies` is a specific augmentation policy, and is composed by several augmentations (dict). When AutoAugment is called, a random policy in `policies` will be selected to augment images.

Parameters `hparams` (dict) – Configs of hyperparameters. Hyperparameters will be used in policies that require these arguments if these arguments are not set in policy dicts. Defaults to use `_HPARAMS_DEFAULT`.

```
__init__(policies=[[{'type': 'Posterize', 'bits': 4, 'prob': 0.4}, {'type': 'Rotate', 'angle': 30.0, 'prob': 0.6}],
                [{'type': 'Solarize', 'thr': 113.77777777777777, 'prob': 0.6}, {'type': 'AutoContrast', 'prob': 0.6}],
                [{'type': 'Equalize', 'prob': 0.8}, {'type': 'Equalize', 'prob': 0.6}], [{'type': 'Posterize', 'bits': 5,
                'prob': 0.6}, {'type': 'Posterize', 'bits': 5, 'prob': 0.6}], [{'type': 'Equalize', 'prob': 0.4}, {'type':
                'Solarize', 'thr': 142.22222222222223, 'prob': 0.2}], [{'type': 'Equalize', 'prob': 0.4}, {'type':
                'Rotate', 'angle': 26.666666666666668, 'prob': 0.8}], [{'type': 'Solarize', 'thr':
                170.66666666666666, 'prob': 0.6}, {'type': 'Equalize', 'prob': 0.6}], [{'type': 'Posterize', 'bits': 6,
                'prob': 0.8}, {'type': 'Equalize', 'prob': 1.0}], [{'type': 'Rotate', 'angle': 10.0, 'prob': 0.2}, {'type':
                'Solarize', 'thr': 28.444444444444443, 'prob': 0.6}], [{'type': 'Equalize', 'prob': 0.6}, {'type':
                'Posterize', 'bits': 5, 'prob': 0.4}], [{'type': 'Rotate', 'angle': 26.666666666666668, 'prob': 0.8},
                {'type': 'ColorTransform', 'magnitude': 0.0, 'prob': 0.4}], [{'type': 'Rotate', 'angle': 30.0, 'prob':
                0.4}, {'type': 'Equalize', 'prob': 0.6}], [{'type': 'Equalize', 'prob': 0.0}, {'type': 'Equalize', 'prob':
                0.8}], [{'type': 'Invert', 'prob': 0.6}, {'type': 'Equalize', 'prob': 1.0}], [{'type': 'ColorTransform',
                'magnitude': 0.4, 'prob': 0.6}, {'type': 'Contrast', 'magnitude': 0.8, 'prob': 1.0}], [{'type': 'Rotate',
                'angle': 26.666666666666668, 'prob': 0.8}, {'type': 'ColorTransform', 'magnitude': 0.2, 'prob':
                1.0}], [{'type': 'ColorTransform', 'magnitude': 0.8, 'prob': 0.8}, {'type': 'Solarize', 'thr':
                56.888888888888886, 'prob': 0.8}], [{'type': 'Sharpness', 'magnitude': 0.7, 'prob': 0.4}, {'type':
                'Invert', 'prob': 0.6}], [{'type': 'Shear', 'magnitude': 0.16666666666666666, 'prob': 0.6, 'direction':
                'horizontal'}, {'type': 'Equalize', 'prob': 1.0}], [{'type': 'ColorTransform', 'magnitude': 0.0, 'prob':
                0.4}, {'type': 'Equalize', 'prob': 0.6}], [{'type': 'Equalize', 'prob': 0.4}, {'type': 'Solarize', 'thr':
                142.22222222222223, 'prob': 0.2}], [{'type': 'Solarize', 'thr': 113.77777777777777, 'prob': 0.6},
                {'type': 'AutoContrast', 'prob': 0.6}], [{'type': 'Invert', 'prob': 0.6}, {'type': 'Equalize', 'prob': 1.0}],
                [{'type': 'ColorTransform', 'magnitude': 0.4, 'prob': 0.6}, {'type': 'Contrast', 'magnitude': 0.8,
                'prob': 1.0}], [{'type': 'Equalize', 'prob': 0.8}, {'type': 'Equalize', 'prob': 0.6}]],
        hparams={'pad_val': 128})
```

Initialize self. See `help(type(self))` for accurate signature.

```

class easycv.datasets.classification.pipelines.auto_augment.MMRandAugment(num_policies,
    magnitude_level,
    magnitude_std=0.0,
    total_level=30,
    policies=[{'type':
        'AutoContrast'},
        {'type': 'Equalize'},
        {'type': 'Invert'},
        {'type': 'Rotate',
        'magnitude_key':
        'angle',
        'magnitude_range':
        (0, 30)}, {'type':
        'Posterize',
        'magnitude_key':
        'bits',
        'magnitude_range':
        (4, 0)}, {'type':
        'Solarize',
        'magnitude_key':
        'thr',
        'magnitude_range':
        (256, 0)}, {'type':
        'SolarizeAdd',
        'magnitude_key':
        'magnitude',
        'magnitude_range':
        (0, 110)}, {'type':
        'ColorTransform',
        'magnitude_key':
        'magnitude',
        'magnitude_range':
        (0, 0.9)}, {'type':
        'Contrast',
        'magnitude_key':
        'magnitude',
        'magnitude_range':
        (0, 0.9)}, {'type':
        'Brightness',
        'magnitude_key':
        'magnitude',
        'magnitude_range':
        (0, 0.9)}, {'type':
        'Sharpness',
        'magnitude_key':
        'magnitude',
        'magnitude_range':
        (0, 0.9)}, {'type':
        'Shear',
        'magnitude_key':
        'magnitude',
        'magnitude_range':
        (0, 0.3), 'direction':
        'horizontal'},
        {'type': 'Shear',
        'magnitude_key':
        'magnitude',
        'magnitude_range':
        (0, 0.3), 'direction':
        'vertical'}, {'type':
        'Shear',
        'magnitude_key':
        'magnitude',
        'magnitude_range':
        (0, 0.3), 'direction':
        'vertical'}],
    magnitude_std=0.0,
    total_level=30,
    num_policies=num_policies,
    magnitude_level=magnitude_level,
    magnitude_std=magnitude_std,
    total_level=total_level)

```

Random augmentation. This data augmentation is proposed in [RandAugment: Practical automated data augmentation with a reduced search space](#). :param policies: The policies of random augmentation. Each

policy in `policies` is one specific augmentation policy (dict). The policy shall at least have key `type`, indicating the type of augmentation. For those which have magnitude, (given to the fact they are named differently in different augmentation,) `magnitude_key` and `magnitude_range` shall be the magnitude argument (str) and the range of magnitude (tuple in the format of (val1, val2)), respectively. Note that val1 is not necessarily less than val2.

Parameters

- **num_policies** (*int*) – Number of policies to select from policies each time.
- **magnitude_level** (*int* / *float*) – Magnitude level for all the augmentation selected.
- **total_level** (*int* / *float*) – Total level for the magnitude. Defaults to 30.
- **magnitude_std** (*Number* / *str*) – Deviation of magnitude noise applied. - If positive number, magnitude is sampled from normal distribution
(mean=magnitude, std=magnitude_std).
 - If 0 or negative number, magnitude remains unchanged.
 - If str “inf”, magnitude is sampled from uniform distribution (range=[min, magnitude]).
- **hparams** (*dict*) – Configs of hyperparameters. Hyperparameters will be used in policies that require these arguments if these arguments are not set in policy dicts. Defaults to use `_HPARAMS_DEFAULT`.

Note: `magnitude_std` will introduce some randomness to policy, modified by <https://github.com/rwightman/pytorch-image-models>. When `magnitude_std=0`, we calculate the magnitude as follows: .. math:

$$\text{magnitude} = \frac{\text{magnitude_level}}{\text{total_level}} \times (\text{val2} - \text{val1}) + \text{val1}$$

```
__init__(num_policies, magnitude_level, magnitude_std=0.0, total_level=30, policies=[{'type':
'AutoContrast'}, {'type': 'Equalize'}, {'type': 'Invert'}, {'type': 'Rotate', 'magnitude_key': 'angle',
'magnitude_range': (0, 30)}, {'type': 'Posterize', 'magnitude_key': 'bits', 'magnitude_range': (4, 0)},
{'type': 'Solarize', 'magnitude_key': 'thr', 'magnitude_range': (256, 0)}, {'type': 'SolarizeAdd',
'magnitude_key': 'magnitude', 'magnitude_range': (0, 110)}, {'type': 'ColorTransform',
'magnitude_key': 'magnitude', 'magnitude_range': (0, 0.9)}, {'type': 'Contrast', 'magnitude_key':
'magnitude', 'magnitude_range': (0, 0.9)}, {'type': 'Brightness', 'magnitude_key': 'magnitude',
'magnitude_range': (0, 0.9)}, {'type': 'Sharpness', 'magnitude_key': 'magnitude',
'magnitude_range': (0, 0.9)}, {'type': 'Shear', 'magnitude_key': 'magnitude', 'magnitude_range': (0,
0.3), 'direction': 'horizontal'}, {'type': 'Shear', 'magnitude_key': 'magnitude', 'magnitude_range':
(0, 0.3), 'direction': 'vertical'}, {'type': 'Translate', 'magnitude_key': 'magnitude',
'magnitude_range': (0, 0.45), 'direction': 'horizontal'}, {'type': 'Translate', 'magnitude_key':
'magnitude', 'magnitude_range': (0, 0.45), 'direction': 'vertical'}], hparams={'pad_val': 128})
```

Initialize self. See help(type(self)) for accurate signature.


```
class easycv.datasets.classification.pipelines.auto_augment.Shear(magnitude, pad_val=128,
                                                                prob=0.5,
                                                                direction='horizontal',
                                                                random_negative_prob=0.5,
                                                                interpolation='bicubic')
```

Bases: object

Shear images. :param *magnitude*: The magnitude used for shear. :type *magnitude*: int | float :param *pad_val*: Pixel *pad_val* value for constant fill.

If a sequence of length 3, it is used to *pad_val* R, G, B channels respectively. Defaults to 128.

Parameters

- **prob** (*float*) – The probability for performing Shear therefore should be in range [0, 1]. Defaults to 0.5.
- **direction** (*str*) – The shearing direction. Options are ‘horizontal’ and ‘vertical’. Defaults to ‘horizontal’.
- **random_negative_prob** (*float*) – The probability that turns the magnitude negative, which should be in range [0,1]. Defaults to 0.5.
- **interpolation** (*str*) – Interpolation method. Options are ‘nearest’, ‘bilinear’, ‘bicubic’, ‘area’, ‘lanczos’. Defaults to ‘bicubic’.

```
__init__(magnitude, pad_val=128, prob=0.5, direction='horizontal', random_negative_prob=0.5,
        interpolation='bicubic')
```

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.classification.pipelines.auto_augment.Translate(magnitude,
                                                                    pad_val=128, prob=0.5,
                                                                    direction='horizontal',
                                                                    ran-
                                                                    dom_negative_prob=0.5,
                                                                    interpolation='nearest')
```

Bases: object

Translate images. :param *magnitude*: The magnitude used for translate. Note that

the offset is calculated by *magnitude* * size in the corresponding direction. With a magnitude of 1, the whole image will be moved out of the range.

Parameters

- **pad_val** (*int*, *Sequence[int]*) – Pixel *pad_val* value for constant fill. If a sequence of length 3, it is used to *pad_val* R, G, B channels respectively. Defaults to 128.
- **prob** (*float*) – The probability for performing translate therefore should be in range [0, 1]. Defaults to 0.5.
- **direction** (*str*) – The translating direction. Options are ‘horizontal’ and ‘vertical’. Defaults to ‘horizontal’.
- **random_negative_prob** (*float*) – The probability that turns the magnitude negative, which should be in range [0,1]. Defaults to 0.5.
- **interpolation** (*str*) – Interpolation method. Options are ‘nearest’, ‘bilinear’, ‘bicubic’, ‘area’, ‘lanczos’. Defaults to ‘nearest’.

```
__init__(magnitude, pad_val=128, prob=0.5, direction='horizontal', random_negative_prob=0.5,
         interpolation='nearest')
```

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.classification.pipelines.auto_augment.Rotate(angle, center=None,
                                                                  scale=1.0, pad_val=128,
                                                                  prob=0.5,
                                                                  random_negative_prob=0.5,
                                                                  interpolation='nearest')
```

Bases: object

Rotate images. :param angle: The angle used for rotate. Positive values stand for clockwise rotation.

Parameters

- **center** (*tuple[float], optional*) – Center point (w, h) of the rotation in the source image. If None, the center of the image will be used. Defaults to None.
- **scale** (*float*) – Isotropic scale factor. Defaults to 1.0.
- **pad_val** (*int, Sequence[int]*) – Pixel pad_val value for constant fill. If a sequence of length 3, it is used to pad_val R, G, B channels respectively. Defaults to 128.
- **prob** (*float*) – The probability for performing Rotate therefore should be in range [0, 1]. Defaults to 0.5.
- **random_negative_prob** (*float*) – The probability that turns the angle negative, which should be in range [0,1]. Defaults to 0.5.
- **interpolation** (*str*) – Interpolation method. Options are 'nearest', 'bilinear', 'bicubic', 'area', 'lanczos'. Defaults to 'nearest'.

```
__init__(angle, center=None, scale=1.0, pad_val=128, prob=0.5, random_negative_prob=0.5,
         interpolation='nearest')
```

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.classification.pipelines.auto_augment.AutoContrast(prob=0.5)
```

Bases: object

Auto adjust image contrast. :param prob: The probability for performing invert therefore should be in range [0, 1]. Defaults to 0.5.

```
__init__(prob=0.5)
```

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.classification.pipelines.auto_augment.Invert(prob=0.5)
```

Bases: object

Invert images. :param prob: The probability for performing invert therefore should be in range [0, 1]. Defaults to 0.5.

```
__init__(prob=0.5)
```

Initialize self. See help(type(self)) for accurate signature.

class easycv.datasets.classification.pipelines.auto_augment.**Equalize**(*prob=0.5*)

Bases: object

Equalize the image histogram. :param prob: The probability for performing invert therefore should be in range [0, 1]. Defaults to 0.5.

__init__(*prob=0.5*)

Initialize self. See help(type(self)) for accurate signature.

class easycv.datasets.classification.pipelines.auto_augment.**Solarize**(*thr, prob=0.5*)

Bases: object

Solarize images (invert all pixel values above a threshold). :param thr: The threshold above which the pixels value will be inverted.

Parameters prob (*float*) – The probability for solarizing therefore should be in range [0, 1]. Defaults to 0.5.

__init__(*thr, prob=0.5*)

Initialize self. See help(type(self)) for accurate signature.

class easycv.datasets.classification.pipelines.auto_augment.**SolarizeAdd**(*magnitude, thr=128, prob=0.5*)

Bases: object

SolarizeAdd images (add a certain value to pixels below a threshold). :param magnitude: The value to be added to pixels below the thr. :type magnitude: int | float :param thr: The threshold below which the pixels value will be

adjusted.

Parameters prob (*float*) – The probability for solarizing therefore should be in range [0, 1]. Defaults to 0.5.

__init__(*magnitude, thr=128, prob=0.5*)

Initialize self. See help(type(self)) for accurate signature.

class easycv.datasets.classification.pipelines.auto_augment.**Posterize**(*bits, prob=0.5*)

Bases: object

Posterize images (reduce the number of bits for each color channel). :param bits: Number of bits for each pixel in the output img,

which should be less or equal to 8.

Parameters prob (*float*) – The probability for posterizing therefore should be in range [0, 1]. Defaults to 0.5.

__init__(*bits, prob=0.5*)

Initialize self. See help(type(self)) for accurate signature.

class easycv.datasets.classification.pipelines.auto_augment.**Contrast**(*magnitude, prob=0.5, random_negative_prob=0.5*)

Bases: object

Adjust images contrast. :param magnitude: The magnitude used for adjusting contrast. A

positive magnitude would enhance the contrast and a negative magnitude would make the image grayer. A magnitude=0 gives the origin img.

Parameters

- **prob** (*float*) – The probability for performing contrast adjusting therefore should be in range [0, 1]. Defaults to 0.5.
- **random_negative_prob** (*float*) – The probability that turns the magnitude negative, which should be in range [0,1]. Defaults to 0.5.

__init__(*magnitude, prob=0.5, random_negative_prob=0.5*)

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.classification.pipelines.auto_augment.ColorTransform(magnitude,  
                                                                           prob=0.5, ran-  
                                                                           dom_negative_prob=0.5)
```

Bases: object

Adjust images color balance. :param magnitude: The magnitude used for color transform. A

positive magnitude would enhance the color and a negative magnitude would make the image grayer. A magnitude=0 gives the origin img.

Parameters

- **prob** (*float*) – The probability for performing ColorTransform therefore should be in range [0, 1]. Defaults to 0.5.
- **random_negative_prob** (*float*) – The probability that turns the magnitude negative, which should be in range [0,1]. Defaults to 0.5.

__init__(*magnitude, prob=0.5, random_negative_prob=0.5*)

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.classification.pipelines.auto_augment.Brightness(magnitude, prob=0.5,  
                                                                           ran-  
                                                                           dom_negative_prob=0.5)
```

Bases: object

Adjust images brightness. :param magnitude: The magnitude used for adjusting brightness. A

positive magnitude would enhance the brightness and a negative magnitude would make the image darker. A magnitude=0 gives the origin img.

Parameters

- **prob** (*float*) – The probability for performing contrast adjusting therefore should be in range [0, 1]. Defaults to 0.5.
- **random_negative_prob** (*float*) – The probability that turns the magnitude negative, which should be in range [0,1]. Defaults to 0.5.

__init__(*magnitude, prob=0.5, random_negative_prob=0.5*)

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.classification.pipelines.auto_augment.Sharpness(magnitude, prob=0.5,  
                                                                    ran-  
                                                                    dom_negative_prob=0.5)
```

Bases: object

Adjust images sharpness. :param magnitude: The magnitude used for adjusting sharpness. A

positive magnitude would enhance the sharpness and a negative magnitude would make the image bulr. A magnitude=0 gives the origin img.

Parameters

- **prob** (*float*) – The probability for performing contrast adjusting therefore should be in range [0, 1]. Defaults to 0.5.
- **random_negative_prob** (*float*) – The probability that turns the magnitude negative, which should be in range [0,1]. Defaults to 0.5.

```
__init__(magnitude, prob=0.5, random_negative_prob=0.5)  
Initialize self. See help(type(self)) for accurate signature.
```

```
class easycv.datasets.classification.pipelines.auto_augment.Cutout(shape, pad_val=128,  
                                                                    prob=0.5)
```

Bases: object

Cutout images. :param shape: Expected cutout shape (h, w).

If given as a single value, the value will be used for both h and w.

Parameters

- **pad_val** (*int, Sequence[int]*) – Pixel pad_val value for constant fill. If it is a sequence, it must have the same length with the image channels. Defaults to 128.
- **prob** (*float*) – The probability for performing cutout therefore should be in range [0, 1]. Defaults to 0.5.

```
__init__(shape, pad_val=128, prob=0.5)  
Initialize self. See help(type(self)) for accurate signature.
```

easycv.datasets.classification.pipelines.transform module

```
class easycv.datasets.classification.pipelines.transform.MMRandomErasing(erase_prob=0.5,  
                                                                    min_area_ratio=0.02,  
                                                                    max_area_ratio=0.4,  
                                                                    aspect_range=(0.3,  
                                                                    3.3333333333333335),  
                                                                    mode='const',  
                                                                    fill_color=(128, 128,  
                                                                    128), fill_std=None)
```

Bases: object

Randomly selects a rectangle region in an image and erase pixels. :param erase_prob: Probability that image will be randomly erased.

Default: 0.5

Parameters

- **min_area_ratio** (*float*) – Minimum erased area / input image area Default: 0.02
- **max_area_ratio** (*float*) – Maximum erased area / input image area Default: 0.4
- **aspect_range** (*sequence | float*) – Aspect ratio range of erased area. if float, it will be converted to (aspect_ratio, 1/aspect_ratio) Default: (3/10, 10/3)
- **mode** (*str*) – Fill method in erased area, can be: - const (default): All pixels are assign with the same value. - rand: each pixel is assigned with a random value in [0, 255]
- **fill_color** (*sequence | Number*) – Base color filled in erased area. Defaults to (128, 128, 128).
- **fill_std** (*sequence | Number, optional*) – If set and mode is 'rand', fill erased area with random color from normal distribution (mean=fill_color, std=fill_std); If not set, fill erased area with random color from uniform distribution (0~255). Defaults to None.

Note: See [Random Erasing Data Augmentation](#) This paper provided 4 modes: RE-R, RE-M, RE-0, RE-255, and use RE-M as default. The config of these 4 modes are: - RE-R: RandomErasing(mode='rand') - RE-M: RandomErasing(mode='const', fill_color=(123.67, 116.3, 103.5)) - RE-0: RandomErasing(mode='const', fill_color=0) - RE-255: RandomErasing(mode='const', fill_color=255)

```
__init__(erase_prob=0.5, min_area_ratio=0.02, max_area_ratio=0.4, aspect_range=(0.3,
3.3333333333333335), mode='const', fill_color=(128, 128, 128), fill_std=None)
Initialize self. See help(type(self)) for accurate signature.
```

Submodules

easycv.datasets.classification.odps module

```
class easycv.datasets.classification.odps.ClsOdpsDataset(data_source, pipeline,
                                                         image_key='url_image',
                                                         label_key='label', **kwargs)
```

Bases: Generic[torch.utils.data.dataset.T_co]

Dataset for rotation prediction

```
__init__(data_source, pipeline, image_key='url_image', label_key='label', **kwargs)
Initialize self. See help(type(self)) for accurate signature.
```

```
evaluate(results, evaluators, logger=None)
```

easycv.datasets.classification.raw module

```
class easycv.datasets.classification.raw.ClsDataset(data_source, pipeline)
```

Bases: Generic[torch.utils.data.dataset.T_co]

Dataset for classification

Parameters

- **data_source** – data source to parse input data
- **pipeline** – transforms list

```
__init__(data_source, pipeline)
Initialize self. See help(type(self)) for accurate signature.
```

evaluate(*results, evaluators, logger=None, topk=(1, 5)*)
 evaluate classification task

Parameters

- **results** – a dict of list of tensor, including prediction and groundtruth info, where prediction tensor is NxCan and the same with groundtruth labels.
- **evaluators** – a list of evaluator

Returns a dict of float, different metric values

Return type eval_result

visualize(*results, vis_num=10, **kwargs*)

Visualize the model output on validation data. :param results: A dictionary containing

class: List of length number of test images. img metas: List of length number of test images, dict of image meta info, containing filename, img_shape, origin_img_shape and so on.

Parameters vis_num – number of images visualized

Returns: A dictionary containing images: Visualized images, list of np.ndarray. img metas: List of length number of test images, dict of image meta info, containing filename, img_shape, origin_img_shape and so on.

12.1.2 easycv.datasets.detection package

class easycv.datasets.detection.DetDataset(*data_source, pipeline, profiling=False, classes=None*)
 Bases: Generic[torch.utils.data.dataset.T_co]

Dataset for Detection

__init__(*data_source, pipeline, profiling=False, classes=None*)

Parameters

- **data_source** – Data_source config dict
- **pipeline** – Pipeline config list
- **profiling** – If set True, will print pipeline time
- **classes** – A list of class names, used in evaluation for result and groundtruth visualization

evaluate(*results, evaluators=None, logger=None*)

Evaluates the detection boxes. :param results: A dictionary containing

detection_boxes: List of length number of test images. Float32 numpy array of shape [num_boxes, 4] and format [ymin, xmin, ymax, xmax] in absolute image coordinates.

detection_scores: List of length number of test images, detection scores for the boxes, float32 numpy array of shape [num_boxes].

detection_classes: List of length number of test images, integer numpy array of shape [num_boxes] containing 1-indexed detection classes for the boxes.

img metas: List of length number of test images, dict of image meta info, containing filename, img_shape, origin_img_shape, scale_factor and so on.

Parameters `evaluators` – evaluators to calculate metric with results and `groundtruth_dict`

visualize(*results*, *vis_num=10*, *score_thr=0.3*, ***kwargs*)

Visualize the model output on validation data. :param results: A dictionary containing

detection_boxes: List of length number of test images. Float32 numpy array of shape [num_boxes, 4] and format [ymin, xmin, ymax, xmax] in absolute image coordinates.

detection_scores: List of length number of test images, detection scores for the boxes, float32 numpy array of shape [num_boxes].

detection_classes: List of length number of test images, integer numpy array of shape [num_boxes] containing 1-indexed detection classes for the boxes.

img metas: List of length number of test images, dict of image meta info, containing filename, img_shape, origin_img_shape, scale_factor and so on.

Parameters

- **vis_num** – number of images visualized
- **score_thr** – The threshold to filter box, boxes with scores greater than `score_thr` will be kept.

Returns: A dictionary containing `images`: Visualized images. `img_metas`: List of length number of test images,

dict of image meta info, containing filename, `img_shape`, `origin_img_shape`, `scale_factor` and so on.

```
class easycv.datasets.detection.DetImagesMixDataset(data_source, pipeline, dynamic_scale=None,  
                                                    skip_type_keys=None, profiling=False,  
                                                    classes=None, yolo_format=True,  
                                                    label_padding=True)
```

Bases: `Generic[torch.utils.data.dataset.T_co]`

A wrapper of multiple images mixed dataset.

Suitable for training on multiple images mixed data augmentation like mosaic and mixup. For the augmentation pipeline of mixed image data, the `get_indexes` method needs to be provided to obtain the image indexes, and you can set `skip_flags` to change the pipeline running process. At the same time, we provide the `dynamic_scale` parameter to dynamically change the output image size.

output boxes format: cx, cy, w, h

Parameters

- **data_source** (`DetSourceCoco`) – The dataset to be mixed.
- **pipeline** (`Sequence[dict]`) – Sequence of transform object or config dict to be composed.
- **dynamic_scale** (`tuple[int]`, *optional*) – The image scale can be changed dynamically. Default to `None`.
- **skip_type_keys** (`list[str]`, *optional*) – Sequence of type string to be skip pipeline. Default to `None`.
- **label_padding** – out labeling padding [N, 120, 5]

__init__(*data_source, pipeline, dynamic_scale=None, skip_type_keys=None, profiling=False, classes=None, yolo_format=True, label_padding=True*)

Args: *data_source*: Data_source config dict *pipeline*: Pipeline config list *profiling*: If set True, will print pipeline time *classes*: A list of class names, used in evaluation for result and groundtruth visualization

update_skip_type_keys(*skip_type_keys*)

Update skip_type_keys. It is called by an external hook.

Parameters *skip_type_keys* (*list[str]*, *optional*) – Sequence of type string to be skip pipeline.

update_dynamic_scale(*dynamic_scale*)

Update dynamic_scale. It is called by an external hook.

Parameters *dynamic_scale* (*tuple[int]*) – The image scale can be changed dynamically.

results2json(*results, outfile_prefix*)

Dump the detection results to a COCO style json file.

There are 3 types of results: proposals, bbox predictions, mask predictions, and they have different data types. This method will automatically recognize the type, and dump them to json files.

Parameters

- **results** (*list[list | tuple | ndarray]*) – Testing results of the dataset.
- **outfile_prefix** (*str*) – The filename prefix of the json files. If the prefix is “somepath/xxx”, the json files will be named “somepath/xxx.bbox.json”, “somepath/xxx.segm.json”, “somepath/xxx.proposal.json”.

Returns *str*: Possible keys are “bbox”, “segm”, “proposal”, and values are corresponding filenames.

Return type *dict[str]*

format_results(*results, jsonfile_prefix=None, **kwargs*)

Format the results to json (standard format for COCO evaluation).

Parameters

- **results** (*list[tuple | numpy.ndarray]*) – Testing results of the dataset.
- **jsonfile_prefix** (*str | None*) – The prefix of json files. It includes the file path and the prefix of filename, e.g., “a/b/prefix”. If not specified, a temp file will be created. Default: None.

Returns (*result_files, tmp_dir*), *result_files* is a dict containing the json filepaths, *tmp_dir* is the temporal directory created for saving json files when *jsonfile_prefix* is not specified.

Return type *tuple*

Subpackages

easycv.datasets.detection.data_sources package

class `easycv.datasets.detection.data_sources.DetSourceCoco`(*ann_file, img_prefix, pipeline, filter_empty_gt=False, classes=None, iscrowd=False*)

Bases: `object`

coco data source

__init__(*ann_file, img_prefix, pipeline, filter_empty_gt=False, classes=None, iscrowd=False*)

Parameters

- **ann_file** – Path of annotation file.
- **img_prefix** – coco path prefix
- **filter_empty_gt** – bool, if filter empty gt
- **iscrowd** – when traing setted as False, when val setted as Tre

get_length()

Total number of samples of data.

load_annotations(*ann_file*)

Load annotation from COCO style annotation file.

Parameters **ann_file** (*str*) – Path of annotation file.

Returns Annotation info from COCO api.

Return type list[dict]

get_ann_info(*idx*)

Get COCO annotation by index.

Parameters **idx** (*int*) – Index of data.

Returns Annotation info of specified index.

Return type dict

get_cat_ids(*idx*)

Get COCO category ids by index.

Parameters **idx** (*int*) – Index of data.

Returns All categories in the image of specified index.

Return type list[int]

xyxy2xywh(*bbox*)

Convert xyxy style bounding boxes to xywh style for COCO evaluation.

Parameters **bbox** (*numpy.ndarray*) – The bounding boxes, shape (4,), in xyxy order.

Returns The converted bounding boxes, in xywh order.

Return type list[float]

pre_pipeline(*results*)

Prepare results dict for pipeline.

prepare_train_img(*idx*)

Get training data and annotations after pipeline.

Parameters **idx** (*int*) – Index of data.

Returns Training data and annotation after pipeline with new keys introduced by pipeline.

Return type dict

get_sample(*idx*)

```
class easycv.datasets.detection.data_sources.DetSourcePAI(path, classes=[], cache_at_init=False,
                                                         cache_on_the_fly=False,
                                                         parse_fn=<function
                                                         parser_manifest_row_str>,
                                                         num_processes=1, **kwargs)
```

Bases: easycv.datasets.detection.data_sources.base.DetSourceBase

data format please refer to: https://help.aliyun.com/document_detail/311173.html

```
__init__(path, classes=[], cache_at_init=False, cache_on_the_fly=False, parse_fn=<function
         parser_manifest_row_str>, num_processes=1, **kwargs)
```

Parameters

- **path** – Path of manifest path with pai label format
- **classes** – classes list
- **cache_at_init** – if set True, will cache in memory in **__init__** for faster training
- **cache_on_the_fly** – if set True, will cache in memroy during training
- **parse_fn** – parse function to parse item of source iterator
- **num_processes** – number of processes to parse samples

get_source_iterator()

Return data list iterator, source iterator will be passed to **parse_fn**, and **parse_fn** will receive params of item of source iter and classes for parsing. What does **parse_fn** need, what does source iterator returns.

```
class easycv.datasets.detection.data_sources.DetSourceRaw(img_root_path, label_root_path,
                                                         classes=[], cache_at_init=False,
                                                         cache_on_the_fly=False, delimiter=' ',
                                                         parse_fn=<function parse_raw>,
                                                         num_processes=1, **kwargs)
```

Bases: easycv.datasets.detection.data_sources.base.DetSourceBase

data dir is as follows: `` `|- data_dir

```
    |-images |-1.jpg |-...
```

```
    |-labels |-1.txt |-...
```

` Label txt file is as follows: The first column is the label id, and columns 2 to 5 are coordinates relative to the image width and height [x_center, y_center, bbox_w, bbox_h]. ` 15 0.519398 0.544087 0.476359 0.572061 2 0.501859 0.820726 0.996281 0.332178 ...
`` .. rubric:: Example

```
data_source = DetSourceRaw( img_root_path='/your/data_dir/images', label_root_path='/your/data_dir/labels',
                             )
```

```
__init__(img_root_path, label_root_path, classes=[], cache_at_init=False, cache_on_the_fly=False,
         delimiter=' ', parse_fn=<function parse_raw>, num_processes=1, **kwargs)
```

Parameters

- **img_root_path** – images dir path
- **label_root_path** – labels dir path
- **classes** (*list, optional*) – classes list

- **cache_at_init** – if set True, will cache in memory in `__init__` for faster training
- **cache_on_the_fly** – if set True, will cache in memroy during training
- **delimiter** – delimiter of txt file
- **parse_fn** – parse function to parse item of source iterator
- **num_processes** – number of processes to parse samples

get_source_iterator()

Return data list iterator, source iterator will be passed to `parse_fn`, and `parse_fn` will receive params of item of source iter and classes for parsing. What does `parse_fn` need, what does source iterator returns.

post_process_fn(result_dict)

```
class easycv.datasets.detection.data_sources.DetSourceVOC(path, classes=[], img_root_path=None,
                                                         label_root_path=None,
                                                         cache_at_init=False,
                                                         cache_on_the_fly=False,
                                                         img_suffix='.jpg', label_suffix='.xml',
                                                         parse_fn=<function parse_xml>,
                                                         num_processes=1, **kwargs)
```

Bases: `easycv.datasets.detection.data_sources.base.DetSourceBase`

data dir is as follows: ``` |- voc_data`

`|-ImageSets`

`|-Main |-train.txt |-...`

`|-JPEGImages |-00001.jpg |-...`

`|-Annotations |-00001.xml |-...`

``` Example1:`

```
data_source = DetSourceVOC(path='/your/voc_data/ImageSets/Main/train.txt',
 classes=${ VOC_CLASSES},
)
```

**Example1:**

```
data_source = DetSourceVOC(path='/your/voc_data/train.txt', classes=${ VOC_CLASSES},
 img_root_path='/your/voc_data/images', img_root_path='/your/voc_data/annotations'
)
```

```
__init__(path, classes=[], img_root_path=None, label_root_path=None, cache_at_init=False,
 cache_on_the_fly=False, img_suffix='.jpg', label_suffix='.xml', parse_fn=<function parse_xml>,
 num_processes=1, **kwargs)
```

#### **Parameters**

- **path** – path of img id list file in ImageSets/Main/
- **classes** – classes list
- **img\_root\_path** – image dir path, if None, default to detect the image dir by the relative path of the `path` according to the VOC data format.
- **label\_root\_path** – label dir path, if None, default to detect the label dir by the relative path of the `path` according to the VOC data format.

- **cache\_at\_init** – if set True, will cache in memory in `__init__` for faster training
- **cache\_on\_the\_fly** – if set True, will cache in memroy during training
- **img\_suffix** – suffix of image file
- **label\_suffix** – suffix of label file
- **parse\_fn** – parse function to parse item of source iterator
- **num\_processes** – number of processes to parse samples

#### **get\_source\_iterator()**

Return data list iterator, source iterator will be passed to `parse_fn`, and `parse_fn` will receive params of item of source iter and classes for parsing. What does `parse_fn` need, what does source iterator returns.

## Submodules

### **easycv.datasets.detection.data\_sources.coco module**

**class** `easycv.datasets.detection.data_sources.coco.DetSourceCoco`(*ann\_file, img\_prefix, pipeline, filter\_empty\_gt=False, classes=None, iscrowd=False*)

Bases: object

coco data source

**\_\_init\_\_**(*ann\_file, img\_prefix, pipeline, filter\_empty\_gt=False, classes=None, iscrowd=False*)

#### **Parameters**

- **ann\_file** – Path of annotation file.
- **img\_prefix** – coco path prefix
- **filter\_empty\_gt** – bool, if filter empty gt
- **iscrowd** – when traing setted as False, when val setted as Tre

#### **get\_length()**

Total number of samples of data.

#### **load\_annotations(ann\_file)**

Load annotation from COCO style annotation file.

**Parameters** **ann\_file** (*str*) – Path of annotation file.

**Returns** Annotation info from COCO api.

**Return type** list[dict]

#### **get\_ann\_info(idx)**

Get COCO annotation by index.

**Parameters** **idx** (*int*) – Index of data.

**Returns** Annotation info of specified index.

**Return type** dict

#### **get\_cat\_ids(idx)**

Get COCO category ids by index.

**Parameters** **idx** (*int*) – Index of data.

**Returns** All categories in the image of specified index.

**Return type** list[int]

**xyxy2xywh**(*bbox*)

Convert xyxy style bounding boxes to xywh style for COCO evaluation.

**Parameters** **bbox** (*numpy.ndarray*) – The bounding boxes, shape (4, ), in xyxy order.

**Returns** The converted bounding boxes, in xywh order.

**Return type** list[float]

**pre\_pipeline**(*results*)

Prepare results dict for pipeline.

**prepare\_train\_img**(*idx*)

Get training data and annotations after pipeline.

**Parameters** **idx** (*int*) – Index of data.

**Returns** Training data and annotation after pipeline with new keys introduced by pipeline.

**Return type** dict

**get\_sample**(*idx*)

## easycv.datasets.detection.data\_sources.pai\_format module

easycv.datasets.detection.data\_sources.pai\_format.**get\_prior\_task\_id**(*keys*)

“The task id ends with *check* is the highest priority.

easycv.datasets.detection.data\_sources.pai\_format.**is\_itag\_v2**(*row*)

The keyword of the data source is *picUrl* in v1, but is *source* in v2

easycv.datasets.detection.data\_sources.pai\_format.**parser\_manifest\_row\_str**(*row\_str*, *classes*)

```
class easycv.datasets.detection.data_sources.pai_format.DetSourcePAI(path, classes=[],
 cache_at_init=False,
 cache_on_the_fly=False,
 parse_fn=<function
 parser_manifest_row_str>,
 num_processes=1,
 **kwargs)
```

Bases: easycv.datasets.detection.data\_sources.base.DetSourceBase

data format please refer to: [https://help.aliyun.com/document\\_detail/311173.html](https://help.aliyun.com/document_detail/311173.html)

```
__init__(path, classes=[], cache_at_init=False, cache_on_the_fly=False, parse_fn=<function
 parser_manifest_row_str>, num_processes=1, **kwargs)
```

### Parameters

- **path** – Path of manifest path with pai label format
- **classes** – classes list
- **cache\_at\_init** – if set True, will cache in memory in `__init__` for faster training
- **cache\_on\_the\_fly** – if set True, will cache in memroy during training
- **parse\_fn** – parse function to parse item of source iterator

- **num\_processes** – number of processes to parse samples

#### **get\_source\_iterator()**

Return data list iterator, source iterator will be passed to `parse_fn`, and `parse_fn` will receive params of item of source iter and classes for parsing. What does `parse_fn` need, what does source iterator returns.

### **easycv.datasets.detection.data\_sources.raw module**

`easycv.datasets.detection.data_sources.raw.parse_raw(source_iter, classes=None, delimiter=' ')`

```
class easycv.datasets.detection.data_sources.raw.DetSourceRaw(img_root_path, label_root_path,
 classes=[], cache_at_init=False,
 cache_on_the_fly=False,
 delimiter=' ', parse_fn=<function
 parse_raw>, num_processes=1,
 **kwargs)
```

Bases: `easycv.datasets.detection.data_sources.base.DetSourceBase`

data dir is as follows: ```` |- data_dir`

`|-images |-1.jpg |-...`

`|-labels |-1.txt |-...`

` Label txt file is as follows: The first column is the label id, and columns 2 to 5 are coordinates relative to the image width and height [x\_center, y\_center, bbox\_w, bbox\_h]. ` 15 0.519398 0.544087 0.476359 0.572061 2 0.501859 0.820726 0.996281 0.332178 ...  
```` .. rubric:: Example`

```
data_source = DetSourceRaw( img_root_path='/your/data_dir/images', label_root_path='/your/data_dir/labels',
                             )
```

```
__init__(img_root_path, label_root_path, classes=[], cache_at_init=False, cache_on_the_fly=False,
          delimiter=' ', parse_fn=<function parse_raw>, num_processes=1, **kwargs)
```

Parameters

- **img_root_path** – images dir path
- **label_root_path** – labels dir path
- **classes** (*list, optional*) – classes list
- **cache_at_init** – if set True, will cache in memory in `__init__` for faster training
- **cache_on_the_fly** – if set True, will cache in memroy during training
- **delimiter** – delimiter of txt file
- **parse_fn** – parse function to parse item of source iterator
- **num_processes** – number of processes to parse samples

get_source_iterator()

Return data list iterator, source iterator will be passed to `parse_fn`, and `parse_fn` will receive params of item of source iter and classes for parsing. What does `parse_fn` need, what does source iterator returns.

post_process_fn(*result_dict*)

easycv.datasets.detection.data_sources.utils module

```
easycv.datasets.detection.data_sources.utils.exif_size(img)
```

easycv.datasets.detection.data_sources.voc module

```
easycv.datasets.detection.data_sources.voc.parse_xml(source_item, classes)
```

```
class easycv.datasets.detection.data_sources.voc.DetSourceVOC(path, classes=[],
                                                             img_root_path=None,
                                                             label_root_path=None,
                                                             cache_at_init=False,
                                                             cache_on_the_fly=False,
                                                             img_suffix='.jpg',
                                                             label_suffix='.xml',
                                                             parse_fn=<function parse_xml>,
                                                             num_processes=1, **kwargs)
```

Bases: easycv.datasets.detection.data_sources.base.DetSourceBase

data dir is as follows: ``` |- voc_data

 |-ImageSets

 |-Main |-train.txt |...

 |-JPEGImages |-00001.jpg |...

 |-Annotations |-00001.xml |...

``` Example1:

```
data_source = DetSourceVOC(path='/your/voc_data/ImageSets/Main/train.txt',
 classes=${ VOC_CLASSES},
)
```

**Example1:**

```
data_source = DetSourceVOC(path='/your/voc_data/train.txt', classes=${ VOC_CLASSES},
 img_root_path='/your/voc_data/images', img_root_path='/your/voc_data/annotations'
)
```

```
__init__(path, classes=[], img_root_path=None, label_root_path=None, cache_at_init=False,
 cache_on_the_fly=False, img_suffix='.jpg', label_suffix='.xml', parse_fn=<function parse_xml>,
 num_processes=1, **kwargs)
```

**Parameters**

- **path** – path of img id list file in ImageSets/Main/
- **classes** – classes list
- **img\_root\_path** – image dir path, if None, default to detect the image dir by the relative path of the *path* according to the VOC data format.
- **label\_root\_path** – label dir path, if None, default to detect the label dir by the relative path of the *path* according to the VOC data format.
- **cache\_at\_init** – if set True, will cache in memory in `__init__` for faster training



- **cache\_on\_the\_fly** – if set True, will cache in memroy during training
- **img\_suffix** – suffix of image file
- **label\_suffix** – suffix of label file
- **parse\_fn** – parse function to parse item of source iterator
- **num\_processes** – number of processes to parse samples

#### **get\_source\_iterator()**

Return data list iterator, source iterator will be passed to `parse_fn`, and `parse_fn` will receive params of item of source iter and classes for parsing. What does `parse_fn` need, what does source iterator returns.

### **easycv.datasets.detection.pipelines package**

#### **class easycv.datasets.detection.pipelines.MMToTensor**

Bases: object

Transform image to Tensor.

Required key: 'img'. Modifies key: 'img'.

**Parameters** **results** (*dict*) – contain all information about training.

#### **class easycv.datasets.detection.pipelines.NormalizeTensor**(*mean, std*)

Bases: object

Normalize the Tensor image (CxHxW), with mean and std.

Required key: 'img'. Modifies key: 'img'.

##### **Parameters**

- **mean** (*list[float]*) – Mean values of 3 channels.
- **std** (*list[float]*) – Std values of 3 channels.

##### **\_\_init\_\_**(*mean, std*)

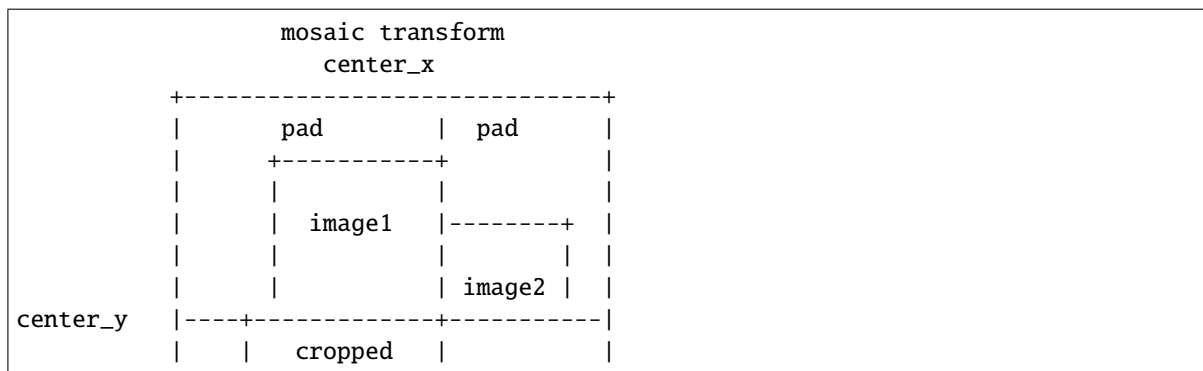
Initialize self. See help(type(self)) for accurate signature.

#### **class easycv.datasets.detection.pipelines.MMMosaic**(*img\_scale=(640, 640), center\_ratio\_range=(0.5, 1.5), pad\_val=114*)

Bases: object

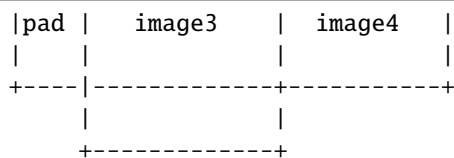
Mosaic augmentation.

Given 4 images, mosaic transform combines them into one output image. The output image is composed of the parts from each sub- image.



(continues on next page)

(continued from previous page)



The mosaic transform steps are as follows:

1. Choose the mosaic center as the intersections of 4 images
2. Get the left top image according to the index, and randomly sample another 3 images from the custom dataset.
3. Sub image will be cropped if image is larger than mosaic patch

### Parameters

- **img\_scale** (*Sequence[int]*) – Image size after mosaic pipeline of single image. Default to (640, 640).
- **center\_ratio\_range** (*Sequence[float]*) – Center ratio range of mosaic output. Default to (0.5, 1.5).
- **pad\_val** (*int*) – Pad value. Default to 114.

**\_\_init\_\_**(*img\_scale=(640, 640), center\_ratio\_range=(0.5, 1.5), pad\_val=114*)

Initialize self. See help(type(self)) for accurate signature.

**get\_indexes**(*dataset*)

Call function to collect indexes.

**Parameters** *dataset* (DetImagesMixDataset) – The dataset.

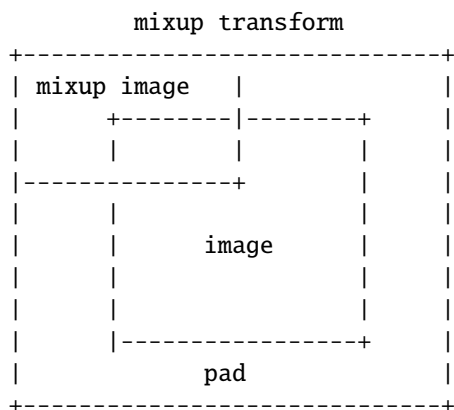
**Returns** indexes.

**Return type** list

**class** easycv.datasets.detection.pipelines.**MMixUp**(*img\_scale=(640, 640), ratio\_range=(0.5, 1.5), flip\_ratio=0.5, pad\_val=114, max\_iters=15, min\_bbox\_size=5, min\_area\_ratio=0.2, max\_aspect\_ratio=20*)

Bases: object

MixUp data augmentation.



(continues on next page)

(continued from previous page)

The mixup transform steps are as follows::

1. Another random image is picked by dataset and embedded in the top left patch(after padding and resizing)
2. The target of mixup transform is the weighted average of mixup image and origin image.

#### Parameters

- **img\_scale** (*Sequence[int]*) – Image output size after mixup pipeline. Default: (640, 640).
- **ratio\_range** (*Sequence[float]*) – Scale ratio of mixup image. Default: (0.5, 1.5).
- **flip\_ratio** (*float*) – Horizontal flip ratio of mixup image. Default: 0.5.
- **pad\_val** (*int*) – Pad value. Default: 114.
- **max\_iters** (*int*) – The maximum number of iterations. If the number of iterations is greater than *max\_iters*, but *gt\_bbox* is still empty, then the iteration is terminated. Default: 15.
- **min\_bbox\_size** (*float*) – Width and height threshold to filter bboxes. If the height or width of a box is smaller than this value, it will be removed. Default: 5.
- **min\_area\_ratio** (*float*) – Threshold of area ratio between original bboxes and wrapped bboxes. If smaller than this value, the box will be removed. Default: 0.2.
- **max\_aspect\_ratio** (*float*) – Aspect ratio of width and height threshold to filter bboxes. If  $\max(h/w, w/h)$  larger than this value, the box will be removed. Default: 20.

**\_\_init\_\_**(*img\_scale=(640, 640), ratio\_range=(0.5, 1.5), flip\_ratio=0.5, pad\_val=114, max\_iters=15, min\_bbox\_size=5, min\_area\_ratio=0.2, max\_aspect\_ratio=20*)

Initialize self. See help(type(self)) for accurate signature.

**get\_indexes**(*dataset*)

Call function to collect indexes.

**Parameters** *dataset* (DetImagesMixDataset) – The dataset.

**Returns** indexes.

**Return type** list

```
class easycv.datasets.detection.pipelines.MMRandomAffine(max_rotate_degree=10.0,
 max_translate_ratio=0.1,
 scaling_ratio_range=(0.5, 1.5),
 max_shear_degree=2.0, border=(0, 0),
 border_val=(114, 114, 114),
 min_bbox_size=2, min_area_ratio=0.2,
 max_aspect_ratio=20)
```

Bases: object

Random affine transform data augmentation. for yolox

This operation randomly generates affine transform matrix which including rotation, translation, shear and scaling transforms.

#### Parameters

- **max\_rotate\_degree** (*float*) – Maximum degrees of rotation transform. Default: 10.

- **max\_translate\_ratio** (*float*) – Maximum ratio of translation. Default: 0.1.
- **scaling\_ratio\_range** (*tuple[*float*]*) – Min and max ratio of scaling transform. Default: (0.5, 1.5).
- **max\_shear\_degree** (*float*) – Maximum degrees of shear transform. Default: 2.
- **border** (*tuple[*int*]*) – Distance from height and width sides of input image to adjust output shape. Only used in mosaic dataset. Default: (0, 0).
- **border\_val** (*tuple[*int*]*) – Border padding values of 3 channels. Default: (114, 114, 114).
- **min\_bbox\_size** (*float*) – Width and height threshold to filter bboxes. If the height or width of a box is smaller than this value, it will be removed. Default: 2.
- **min\_area\_ratio** (*float*) – Threshold of area ratio between original bboxes and wrapped bboxes. If smaller than this value, the box will be removed. Default: 0.2.
- **max\_aspect\_ratio** (*float*) – Aspect ratio of width and height threshold to filter bboxes. If max(h/w, w/h) larger than this value, the box will be removed.

```
__init__(max_rotate_degree=10.0, max_translate_ratio=0.1, scaling_ratio_range=(0.5, 1.5),
 max_shear_degree=2.0, border=(0, 0), border_val=(114, 114, 114), min_bbox_size=2,
 min_area_ratio=0.2, max_aspect_ratio=20)
```

Initialize self. See help(type(self)) for accurate signature.

```
filter_gt_bboxes(origin_bboxes, wrapped_bboxes)
```

```
class easycv.datasets.detection.pipelines.MMPhotoMetricDistortion(brightness_delta=32,
 contrast_range=(0.5, 1.5),
 saturation_range=(0.5, 1.5),
 hue_delta=18)
```

Bases: object

Apply photometric distortion to image sequentially, every transformation is applied with a probability of 0.5. The position of random contrast is in second or second to last.

1. random brightness
2. random contrast (mode 0)
3. convert color from BGR to HSV
4. random saturation
5. random hue
6. convert color from HSV to BGR
7. random contrast (mode 1)
8. randomly swap channels

#### Parameters

- **brightness\_delta** (*int*) – delta of brightness.
- **contrast\_range** (*tuple*) – range of contrast.
- **saturation\_range** (*tuple*) – range of saturation.
- **hue\_delta** (*int*) – delta of hue.

```
__init__(brightness_delta=32, contrast_range=(0.5, 1.5), saturation_range=(0.5, 1.5), hue_delta=18)
 Initialize self. See help(type(self)) for accurate signature.
```

```
class easycv.datasets.detection.pipelines.MMResize(img_scale=None, multiscale_mode='range',
 ratio_range=None, keep_ratio=True,
 bbox_clip_border=True, backend='cv2',
 override=False)
```

Bases: object

Resize images & bbox & mask.

This transform resizes the input image to some scale. Bboxes and masks are then resized with the same scale factor. If the input dict contains the key “scale”, then the scale in the input dict is used, otherwise the specified scale in the init method is used. If the input dict contains the key “scale\_factor” (if MultiScaleFlipAug does not give img\_scale but scale\_factor), the actual scale will be computed by image shape and scale\_factor.

*img\_scale* can either be a tuple (single-scale) or a list of tuple (multi-scale). There are 3 multiscale modes:

- **ratio\_range** is not None: randomly sample a ratio from the ratio range and multiply it with the image scale.
- **ratio\_range** is None and **multiscale\_mode** == “range”: randomly sample a scale from the multi-scale range.
- **ratio\_range** is None and **multiscale\_mode** == “value”: randomly sample a scale from multiple scales.

#### Parameters

- **img\_scale** (*tuple or list[tuple]*) – Images scales for resizing.
- **multiscale\_mode** (*str*) – Either “range” or “value”.
- **ratio\_range** (*tuple[float]*) – (min\_ratio, max\_ratio)
- **keep\_ratio** (*bool*) – Whether to keep the aspect ratio when resizing the image.
- **bbox\_clip\_border** (*bool, optional*) – Whether clip the objects outside the border of the image. Defaults to True.
- **backend** (*str*) – Image resize backend, choices are ‘cv2’ and ‘pillow’. These two backends generates slightly different results. Defaults to ‘cv2’.
- **override** (*bool, optional*) – Whether to override *scale* and *scale\_factor* so as to call resize twice. Default False. If True, after the first resizing, the existed *scale* and *scale\_factor* will be ignored so the second resizing can be allowed. This option is a work-around for multiple times of resize in DETR. Defaults to False.

```
__init__(img_scale=None, multiscale_mode='range', ratio_range=None, keep_ratio=True,
 bbox_clip_border=True, backend='cv2', override=False)
 Initialize self. See help(type(self)) for accurate signature.
```

```
static random_select(img_scales)
 Randomly select an img_scale from given candidates.
```

**Parameters** **img\_scales** (*list[tuple]*) – Images scales for selection.

**Returns** Returns a tuple (*img\_scale*, *scale\_idx*), where *img\_scale* is the selected image scale and *scale\_idx* is the selected index in the given candidates.

**Return type** (tuple, int)

**static random\_sample(*img\_scales*)**

Randomly sample an *img\_scale* when *multiscale\_mode*=='range'.

**Parameters** *img\_scales* (*list[tuple]*) – Images scale range for sampling. There must be two tuples in *img\_scales*, which specify the lower and upper bound of image scales.

**Returns** Returns a tuple (*img\_scale*, *None*), where *img\_scale* is sampled scale and *None* is just a placeholder to be consistent with [random\\_select\(\)](#).

**Return type** (tuple, None)

**static random\_sample\_ratio(*img\_scale*, *ratio\_range*)**

Randomly sample an *img\_scale* when *ratio\_range* is specified.

A ratio will be randomly sampled from the range specified by *ratio\_range*. Then it would be multiplied with *img\_scale* to generate sampled scale.

**Parameters**

- ***img\_scale*** (*tuple*) – Images scale base to multiply with ratio.
- ***ratio\_range*** (*tuple[float]*) – The minimum and maximum ratio to scale the *img\_scale*.

**Returns** Returns a tuple (*scale*, *None*), where *scale* is sampled ratio multiplied with *img\_scale* and *None* is just a placeholder to be consistent with [random\\_select\(\)](#).

**Return type** (tuple, None)

**class easycv.datasets.detection.pipelines.MMRandomFlip(*flip\_ratio*=None, *direction*='horizontal')**

Bases: object

Flip the image & bbox & mask.

If the input dict contains the key “flip”, then the flag will be used, otherwise it will be randomly decided by a ratio specified in the init method.

When random flip is enabled, *flip\_ratio*/*direction* can either be a float/string or tuple of float/string. There are 3 flip modes:

- ***flip\_ratio* is float, *direction* is string:** the image will be *direction*’ly flipped with probability of *flip\_ratio*. E.g., *flip\_ratio*=0.5, *direction*='horizontal', then image will be horizontally flipped with probability of 0.5.
- ***flip\_ratio* is float, *direction* is list of string:** the image will be *direction*[*i*]’ly flipped with probability of *flip\_ratio*/*len(direction)*. E.g., *flip\_ratio*=0.5, *direction*=['horizontal', 'vertical'], then image will be horizontally flipped with probability of 0.25, vertically with probability of 0.25.
- ***flip\_ratio* is list of float, *direction* is list of string:** given *len(flip\_ratio) == len(direction)*, the image will be *direction*[*i*]’ly flipped with probability of *flip\_ratio*[*i*]. E.g., *flip\_ratio*=[0.3, 0.5], *direction*=['horizontal', 'vertical'], then image will be horizontally flipped with probability of 0.3, vertically with probability of 0.5.

**Parameters**

- ***flip\_ratio*** (*float | list[float]*, *optional*) – The flipping probability. Default: *None*.
- ***direction*** (*str | list[str]*, *optional*) – The flipping direction. Options are ‘horizontal’, ‘vertical’, ‘diagonal’. Default: ‘horizontal’. If input is a list, the length must equal

**flip\_ratio.** Each element in **flip\_ratio** indicates the flip probability of corresponding direction.

**\_\_init\_\_**(*flip\_ratio=None, direction='horizontal'*)  
Initialize self. See help(type(self)) for accurate signature.

**bbox\_flip**(*bboxes, img\_shape, direction*)  
Flip bboxes horizontally.

#### Parameters

- **bboxes** (*numpy.ndarray*) – Bounding boxes, shape (... , 4\*k)
- **img\_shape** (*tuple[int]*) – Image shape (height, width)
- **direction** (*str*) – Flip direction. Options are 'horizontal', 'vertical'.

**Returns** Flipped bounding boxes.

**Return type** *numpy.ndarray*

**class** *easycv.datasets.detection.pipelines.MMPad*(*size=None, size\_divisor=None, pad\_to\_square=False, pad\_val=0*)

Bases: object

Pad the image & mask.

There are two padding modes: (1) pad to a fixed size and (2) pad to the minimum size that is divisible by some number. Added keys are "pad\_shape", "pad\_fixed\_size", "pad\_size\_divisor",

#### Parameters

- **size** (*tuple, optional*) – Fixed padding size.
- **size\_divisor** (*int, optional*) – The divisor of padded size.
- **pad\_to\_square** (*bool*) – Whether to pad the image into a square. Currently only used for YOLOX. Default: False.
- **pad\_val** (*float, optional*) – Padding value, 0 by default.

**\_\_init\_\_**(*size=None, size\_divisor=None, pad\_to\_square=False, pad\_val=0*)  
Initialize self. See help(type(self)) for accurate signature.

**class** *easycv.datasets.detection.pipelines.MMNormalize*(*mean, std, to\_rgb=True*)

Bases: object

Normalize the image.

Added key is "img\_norm\_cfg".

#### Parameters

- **mean** (*sequence*) – Mean values of 3 channels.
- **std** (*sequence*) – Std values of 3 channels.
- **to\_rgb** (*bool*) – Whether to convert the image from BGR to RGB, default is true.

**\_\_init\_\_**(*mean, std, to\_rgb=True*)  
Initialize self. See help(type(self)) for accurate signature.

**class** *easycv.datasets.detection.pipelines.LoadImageFromFile*(*to\_float32=False, color\_type='color', file\_client\_args={'backend': 'disk'}*)

Bases: object

Load an image from file.

Required keys are “img\_prefix” and “img\_info” (a dict that must contain the key “filename”). Added or updated keys are “filename”, “img”, “img\_shape”, “ori\_shape” (same as *img\_shape*), “pad\_shape” (same as *img\_shape*), “scale\_factor” (1.0) and “img\_norm\_cfg” (means=0 and stds=1).

#### Parameters

- **to\_float32** (*bool*) – Whether to convert the loaded image to a float32 numpy array. If set to False, the loaded image is an uint8 array. Defaults to False.
- **color\_type** (*str*) – The flag argument for `mmcv.imfrombytes()`. Defaults to ‘color’.
- **file\_client\_args** (*dict*) – Arguments to instantiate a FileClient. See `mmcv.fileio.FileClient` for details. Defaults to `dict(backend='disk')`.

**\_\_init\_\_**(*to\_float32=False, color\_type='color', file\_client\_args={'backend': 'disk'})*  
Initialize self. See `help(type(self))` for accurate signature.

```
class easycv.datasets.detection.pipelines.LoadMultiChannelImageFromFiles(to_float32=False,
 color_type='unchanged',
 file_client_args={'backend':
 'disk'})
```

Bases: object

Load multi-channel images from a list of separate channel files.

Required keys are “img\_prefix” and “img\_info” (a dict that must contain the key “filename”, which is expected to be a list of filenames). Added or updated keys are “filename”, “img”, “img\_shape”, “ori\_shape” (same as *img\_shape*), “pad\_shape” (same as *img\_shape*), “scale\_factor” (1.0) and “img\_norm\_cfg” (means=0 and stds=1).

#### Parameters

- **to\_float32** (*bool*) – Whether to convert the loaded image to a float32 numpy array. If set to False, the loaded image is an uint8 array. Defaults to False.
- **color\_type** (*str*) – The flag argument for `mmcv.imfrombytes()`. Defaults to ‘color’.
- **file\_client\_args** (*dict*) – Arguments to instantiate a FileClient. See `mmcv.fileio.FileClient` for details. Defaults to `dict(backend='disk')`.

**\_\_init\_\_**(*to\_float32=False, color\_type='unchanged', file\_client\_args={'backend': 'disk'})*  
Initialize self. See `help(type(self))` for accurate signature.

```
class easycv.datasets.detection.pipelines.LoadAnnotations(with_bbox=True, with_label=True,
 with_mask=False, with_seg=False,
 poly2mask=True,
 file_client_args={'backend': 'disk'})
```

Bases: object

Load multiple types of annotations.

#### Parameters

- **with\_bbox** (*bool*) – Whether to parse and load the bbox annotation. Default: True.
- **with\_label** (*bool*) – Whether to parse and load the label annotation. Default: True.
- **with\_mask** (*bool*) – Whether to parse and load the mask annotation. Default: False.
- **with\_seg** (*bool*) – Whether to parse and load the semantic segmentation annotation. Default: False.
- **poly2mask** (*bool*) – Whether to convert the instance masks from polygons to bitmaps. Default: True.



- **file\_client\_args** (*dict*) – Arguments to instantiate a FileClient. See `mmcv.fileio.FileClient` for details. Defaults to `dict(backend='disk')`.

**\_\_init\_\_** (*with\_bbox=True, with\_label=True, with\_mask=False, with\_seg=False, poly2mask=True, file\_client\_args={'backend': 'disk'}*)

Initialize self. See `help(type(self))` for accurate signature.

**process\_polygons** (*polygons*)

Convert polygons to list of ndarray and filter invalid polygons.

**Parameters** **polygons** (*list[list]*) – Polygons of one instance.

**Returns** Processed polygons.

**Return type** `list[numpy.ndarray]`

**class** `easycv.datasets.detection.pipelines.MMMultiScaleFlipAug` (*transforms, img\_scale=None, scale\_factor=None, flip=False, flip\_direction='horizontal'*)

Bases: `object`

Test-time augmentation with multiple scales and flipping.

An example configuration is as followed:

```
img_scale=[(1333, 400), (1333, 800)],
flip=True,
transforms=[
 dict(type='Resize', keep_ratio=True),
 dict(type='RandomFlip'),
 dict(type='Normalize', **img_norm_cfg),
 dict(type='Pad', size_divisor=32),
 dict(type='ImageToTensor', keys=['img']),
 dict(type='Collect', keys=['img']),
]
```

After `MultiScaleFlipAug` with above configuration, the results are wrapped into lists of the same length as followed:

```
dict(
 img=[...],
 img_shape=[...],
 scale=[(1333, 400), (1333, 400), (1333, 800), (1333, 800)]
 flip=[False, True, False, True]
 ...
)
```

### Parameters

- **transforms** (*list[dict]*) – Transforms to apply in each augmentation.
- **img\_scale** (*tuple | list[tuple] | None*) – Images scales for resizing.
- **scale\_factor** (*float | list[float] | None*) – Scale factors for resizing.
- **flip** (*bool*) – Whether apply flip augmentation. Default: `False`.
- **flip\_direction** (*str | list[str]*) – Flip augmentation directions, options are “horizontal”, “vertical” and “diagonal”. If `flip_direction` is a list, multiple flip augmentations will be applied. It has no effect when `flip == False`. Default: “horizontal”.

**\_\_init\_\_**(*transforms, img\_scale=None, scale\_factor=None, flip=False, flip\_direction='horizontal'*)  
 Initialize self. See help(type(self)) for accurate signature.

## Submodules

### easycv.datasets.detection.pipelines.mm\_transforms module

**class** easycv.datasets.detection.pipelines.mm\_transforms.**MMToTensor**

Bases: object

Transform image to Tensor.

Required key: 'img'. Modifies key: 'img'.

**Parameters results** (*dict*) – contain all information about training.

**class** easycv.datasets.detection.pipelines.mm\_transforms.**NormalizeTensor**(*mean, std*)

Bases: object

Normalize the Tensor image (CxHxW), with mean and std.

Required key: 'img'. Modifies key: 'img'.

#### Parameters

- **mean** (*list[float]*) – Mean values of 3 channels.
- **std** (*list[float]*) – Std values of 3 channels.

**\_\_init\_\_**(*mean, std*)

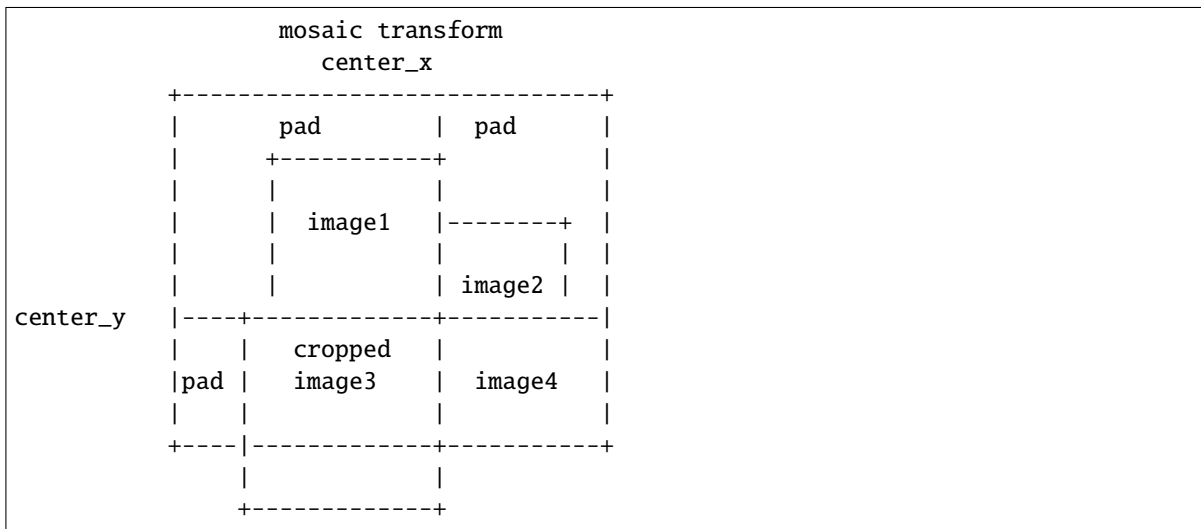
Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.detection.pipelines.mm\_transforms.**MMMosaic**(*img\_scale=(640, 640), center\_ratio\_range=(0.5, 1.5), pad\_val=114*)

Bases: object

Mosaic augmentation.

Given 4 images, mosaic transform combines them into one output image. The output image is composed of the parts from each sub- image.



(continues on next page)

(continued from previous page)

The mosaic transform steps are as follows:

1. Choose the mosaic center as the intersections of 4 images
2. Get the left top image according to the index, and randomly sample another 3 images from the custom dataset.
3. Sub image will be cropped if image is larger than mosaic patch

#### Parameters

- **img\_scale** (*Sequence[int]*) – Image size after mosaic pipeline of single image. Default to (640, 640).
- **center\_ratio\_range** (*Sequence[float]*) – Center ratio range of mosaic output. Default to (0.5, 1.5).
- **pad\_val** (*int*) – Pad value. Default to 114.

**\_\_init\_\_** (*img\_scale=(640, 640), center\_ratio\_range=(0.5, 1.5), pad\_val=114*)

Initialize self. See help(type(self)) for accurate signature.

**get\_indexes** (*dataset*)

Call function to collect indexes.

**Parameters** *dataset* (DetImagesMixDataset) – The dataset.

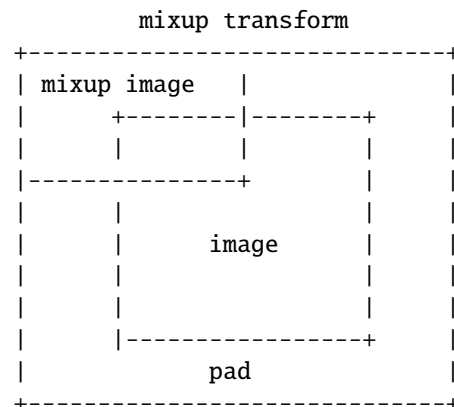
**Returns** indexes.

**Return type** list

```
class easycv.datasets.detection.pipelines.mmm_transforms.MMMixUp(img_scale=(640, 640),
 ratio_range=(0.5, 1.5),
 flip_ratio=0.5, pad_val=114,
 max_iters=15,
 min_bbox_size=5,
 min_area_ratio=0.2,
 max_aspect_ratio=20)
```

Bases: object

MixUp data augmentation.



The mixup transform steps are as follows::

(continues on next page)

(continued from previous page)

1. Another random image is picked by dataset and embedded in the top left patch(after padding and resizing)
2. The target of mixup transform is the weighted average of mixup image and origin image.

**Parameters**

- **img\_scale** (*Sequence[int]*) – Image output size after mixup pipeline. Default: (640, 640).
- **ratio\_range** (*Sequence[float]*) – Scale ratio of mixup image. Default: (0.5, 1.5).
- **flip\_ratio** (*float*) – Horizontal flip ratio of mixup image. Default: 0.5.
- **pad\_val** (*int*) – Pad value. Default: 114.
- **max\_iters** (*int*) – The maximum number of iterations. If the number of iterations is greater than *max\_iters*, but *gt\_bbox* is still empty, then the iteration is terminated. Default: 15.
- **min\_bbox\_size** (*float*) – Width and height threshold to filter bboxes. If the height or width of a box is smaller than this value, it will be removed. Default: 5.
- **min\_area\_ratio** (*float*) – Threshold of area ratio between original bboxes and wrapped bboxes. If smaller than this value, the box will be removed. Default: 0.2.
- **max\_aspect\_ratio** (*float*) – Aspect ratio of width and height threshold to filter bboxes. If  $\max(h/w, w/h)$  larger than this value, the box will be removed. Default: 20.

**\_\_init\_\_** (*img\_scale=(640, 640), ratio\_range=(0.5, 1.5), flip\_ratio=0.5, pad\_val=114, max\_iters=15, min\_bbox\_size=5, min\_area\_ratio=0.2, max\_aspect\_ratio=20*)

Initialize self. See help(type(self)) for accurate signature.

**get\_indexes** (*dataset*)

Call function to collect indexes.

**Parameters** *dataset* (*DetImagesMixDataset*) – The dataset.

**Returns** indexes.

**Return type** list

```
class easycv.datasets.detection.pipelines.mmm_transforms.MMRandomAffine(max_rotate_degree=10.0,
 max_translate_ratio=0.1,
 scaling_ratio_range=(0.5,
 1.5),
 max_shear_degree=2.0,
 border=(0, 0),
 border_val=(114, 114,
 114), min_bbox_size=2,
 min_area_ratio=0.2,
 max_aspect_ratio=20)
```

Bases: object

Random affine transform data augmentation. for yolox

This operation randomly generates affine transform matrix which including rotation, translation, shear and scaling transforms.

**Parameters**

- **max\_rotate\_degree** (*float*) – Maximum degrees of rotation transform. Default: 10.
- **max\_translate\_ratio** (*float*) – Maximum ratio of translation. Default: 0.1.
- **scaling\_ratio\_range** (*tuple[*float*]*) – Min and max ratio of scaling transform. Default: (0.5, 1.5).
- **max\_shear\_degree** (*float*) – Maximum degrees of shear transform. Default: 2.
- **border** (*tuple[*int*]*) – Distance from height and width sides of input image to adjust output shape. Only used in mosaic dataset. Default: (0, 0).
- **border\_val** (*tuple[*int*]*) – Border padding values of 3 channels. Default: (114, 114, 114).
- **min\_bbox\_size** (*float*) – Width and height threshold to filter bboxes. If the height or width of a box is smaller than this value, it will be removed. Default: 2.
- **min\_area\_ratio** (*float*) – Threshold of area ratio between original bboxes and wrapped bboxes. If smaller than this value, the box will be removed. Default: 0.2.
- **max\_aspect\_ratio** (*float*) – Aspect ratio of width and height threshold to filter bboxes. If max(h/w, w/h) larger than this value, the box will be removed.

```
__init__(max_rotate_degree=10.0, max_translate_ratio=0.1, scaling_ratio_range=(0.5, 1.5),
 max_shear_degree=2.0, border=(0, 0), border_val=(114, 114, 114), min_bbox_size=2,
 min_area_ratio=0.2, max_aspect_ratio=20)
```

Initialize self. See help(type(self)) for accurate signature.

```
filter_gt_bboxes(origin_bboxes, wrapped_bboxes)
```

```
class easycv.datasets.detection.pipelines.mm_transforms.MMPhotoMetricDistortion(brightness_delta=32,
 con-
 trast_range=(0.5,
 1.5), satura-
 tion_range=(0.5,
 1.5),
 hue_delta=18)
```

Bases: object

Apply photometric distortion to image sequentially, every transformation is applied with a probability of 0.5. The position of random contrast is in second or second to last.

1. random brightness
2. random contrast (mode 0)
3. convert color from BGR to HSV
4. random saturation
5. random hue
6. convert color from HSV to BGR
7. random contrast (mode 1)
8. randomly swap channels

**Parameters**

- **brightness\_delta** (*int*) – delta of brightness.

- **contrast\_range** (*tuple*) – range of contrast.
- **saturation\_range** (*tuple*) – range of saturation.
- **hue\_delta** (*int*) – delta of hue.

**\_\_init\_\_** (*brightness\_delta=32, contrast\_range=(0.5, 1.5), saturation\_range=(0.5, 1.5), hue\_delta=18*)

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.detection.pipelines.mm_transforms.MMResize(img_scale=None,
 multiscale_mode='range',
 ratio_range=None,
 keep_ratio=True,
 bbox_clip_border=True,
 backend='cv2',
 override=False)
```

Bases: object

Resize images & bbox & mask.

This transform resizes the input image to some scale. Bboxes and masks are then resized with the same scale factor. If the input dict contains the key “scale”, then the scale in the input dict is used, otherwise the specified scale in the init method is used. If the input dict contains the key “scale\_factor” (if MultiScaleFlipAug does not give *img\_scale* but *scale\_factor*), the actual scale will be computed by image shape and *scale\_factor*.

*img\_scale* can either be a tuple (single-scale) or a list of tuple (multi-scale). There are 3 multiscale modes:

- **ratio\_range** is not None: randomly sample a ratio from the ratio range and multiply it with the image scale.
- **ratio\_range** is None and **multiscale\_mode** == “range”: randomly sample a scale from the multi-scale range.
- **ratio\_range** is None and **multiscale\_mode** == “value”: randomly sample a scale from multiple scales.

#### Parameters

- **img\_scale** (*tuple or list[tuple]*) – Images scales for resizing.
- **multiscale\_mode** (*str*) – Either “range” or “value”.
- **ratio\_range** (*tuple[float]*) – (min\_ratio, max\_ratio)
- **keep\_ratio** (*bool*) – Whether to keep the aspect ratio when resizing the image.
- **bbox\_clip\_border** (*bool, optional*) – Whether clip the objects outside the border of the image. Defaults to True.
- **backend** (*str*) – Image resize backend, choices are ‘cv2’ and ‘pillow’. These two backends generates slightly different results. Defaults to ‘cv2’.
- **override** (*bool, optional*) – Whether to override *scale* and *scale\_factor* so as to call resize twice. Default False. If True, after the first resizing, the existed *scale* and *scale\_factor* will be ignored so the second resizing can be allowed. This option is a work-around for multiple times of resize in DETR. Defaults to False.

**\_\_init\_\_** (*img\_scale=None, multiscale\_mode='range', ratio\_range=None, keep\_ratio=True,*  
 *bbox\_clip\_border=True, backend='cv2', override=False*)

Initialize self. See help(type(self)) for accurate signature.

**static random\_select**(*img\_scales*)

Randomly select an *img\_scale* from given candidates.

**Parameters** *img\_scales* (*list[tuple]*) – Images scales for selection.

**Returns** Returns a tuple (*img\_scale*, *scale\_idx*), where *img\_scale* is the selected image scale and *scale\_idx* is the selected index in the given candidates.

**Return type** (tuple, int)

**static random\_sample**(*img\_scales*)

Randomly sample an *img\_scale* when *multiscale\_mode*=='range'.

**Parameters** *img\_scales* (*list[tuple]*) – Images scale range for sampling. There must be two tuples in *img\_scales*, which specify the lower and upper bound of image scales.

**Returns** Returns a tuple (*img\_scale*, None), where *img\_scale* is sampled scale and None is just a placeholder to be consistent with [random\\_select\(\)](#).

**Return type** (tuple, None)

**static random\_sample\_ratio**(*img\_scale*, *ratio\_range*)

Randomly sample an *img\_scale* when *ratio\_range* is specified.

A ratio will be randomly sampled from the range specified by *ratio\_range*. Then it would be multiplied with *img\_scale* to generate sampled scale.

**Parameters**

- **img\_scale** (*tuple*) – Images scale base to multiply with ratio.
- **ratio\_range** (*tuple[float]*) – The minimum and maximum ratio to scale the *img\_scale*.

**Returns** Returns a tuple (*scale*, None), where *scale* is sampled ratio multiplied with *img\_scale* and None is just a placeholder to be consistent with [random\\_select\(\)](#).

**Return type** (tuple, None)

**class** `easycv.datasets.detection.pipelines.mm_transforms.MMRandomFlip`(*flip\_ratio=None*,  
*direction='horizontal'*)

Bases: object

Flip the image & bbox & mask.

If the input dict contains the key “flip”, then the flag will be used, otherwise it will be randomly decided by a ratio specified in the init method.

When random flip is enabled, *flip\_ratio*/*direction* can either be a float/string or tuple of float/string. There are 3 flip modes:

- **flip\_ratio is float, direction is string:** the image will be *direction*’ly flipped with probability of *flip\_ratio*. E.g., *flip\_ratio*=0.5, *direction*='horizontal', then image will be horizontally flipped with probability of 0.5.
- **flip\_ratio is float, direction is list of string:** the image will be *direction[i]*’ly flipped with probability of *flip\_ratio*/*len(direction)*. E.g., *flip\_ratio*=0.5, *direction*=['horizontal', 'vertical'], then image will be horizontally flipped with probability of 0.25, vertically with probability of 0.25.
- **flip\_ratio is list of float, direction is list of string:** given *len(flip\_ratio) == len(direction)*, the image will be *direction[i]*’ly flipped with probability of *flip\_ratio[i]*. E.g., *flip\_ratio*=[0.3, 0.5], *direction*=['horizontal',

'vertical'], then image will be horizontally flipped with probability of 0.3, vertically with probability of 0.5.

#### Parameters

- **flip\_ratio** (*float | list[float], optional*) – The flipping probability. Default: None.
- **direction** (*str | list[str], optional*) – The flipping direction. Options are 'horizontal', 'vertical', 'diagonal'. Default: 'horizontal'. If input is a list, the length must equal flip\_ratio. Each element in flip\_ratio indicates the flip probability of corresponding direction.

**\_\_init\_\_** (*flip\_ratio=None, direction='horizontal'*)

Initialize self. See help(type(self)) for accurate signature.

**bbox\_flip** (*bboxes, img\_shape, direction*)

Flip bboxes horizontally.

#### Parameters

- **bboxes** (*numpy.ndarray*) – Bounding boxes, shape (... , 4\*k)
- **img\_shape** (*tuple[int]*) – Image shape (height, width)
- **direction** (*str*) – Flip direction. Options are 'horizontal', 'vertical'.

**Returns** Flipped bounding boxes.

**Return type** *numpy.ndarray*

**class** *easycv.datasets.detection.pipelines.mm\_transforms.MMPad* (*size=None, size\_divisor=None, pad\_to\_square=False, pad\_val=0*)

Bases: *object*

Pad the image & mask.

There are two padding modes: (1) pad to a fixed size and (2) pad to the minimum size that is divisible by some number. Added keys are "pad\_shape", "pad\_fixed\_size", "pad\_size\_divisor",

#### Parameters

- **size** (*tuple, optional*) – Fixed padding size.
- **size\_divisor** (*int, optional*) – The divisor of padded size.
- **pad\_to\_square** (*bool*) – Whether to pad the image into a square. Currently only used for YOLOX. Default: False.
- **pad\_val** (*float, optional*) – Padding value, 0 by default.

**\_\_init\_\_** (*size=None, size\_divisor=None, pad\_to\_square=False, pad\_val=0*)

Initialize self. See help(type(self)) for accurate signature.

**class** *easycv.datasets.detection.pipelines.mm\_transforms.MMNormalize* (*mean, std, to\_rgb=True*)

Bases: *object*

Normalize the image.

Added key is "img\_norm\_cfg".

#### Parameters

- **mean** (*sequence*) – Mean values of 3 channels.
- **std** (*sequence*) – Std values of 3 channels.



- **to\_rgb** (*bool*) – Whether to convert the image from BGR to RGB, default is true.

**\_\_init\_\_** (*mean, std, to\_rgb=True*)

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.detection.pipelines.mm_transforms.LoadImageFromFile(to_float32=False,
 color_type='color',
 file_client_args={'backend':
 'disk'})
```

Bases: object

Load an image from file.

Required keys are “img\_prefix” and “img\_info” (a dict that must contain the key “filename”). Added or updated keys are “filename”, “img”, “img\_shape”, “ori\_shape” (same as *img\_shape*), “pad\_shape” (same as *img\_shape*), “scale\_factor” (1.0) and “img\_norm\_cfg” (means=0 and stds=1).

#### Parameters

- **to\_float32** (*bool*) – Whether to convert the loaded image to a float32 numpy array. If set to False, the loaded image is an uint8 array. Defaults to False.
- **color\_type** (*str*) – The flag argument for `mmcv.imfrombytes()`. Defaults to ‘color’.
- **file\_client\_args** (*dict*) – Arguments to instantiate a FileClient. See `mmcv.fileio.FileClient` for details. Defaults to `dict(backend='disk')`.

**\_\_init\_\_** (*to\_float32=False, color\_type='color', file\_client\_args={'backend': 'disk'}*)

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.detection.pipelines.mm_transforms.LoadMultiChannelImageFromFiles(to_float32=False,
 color_type='unchang
 file_client_args={'ba
 'disk'})
```

Bases: object

Load multi-channel images from a list of separate channel files.

Required keys are “img\_prefix” and “img\_info” (a dict that must contain the key “filename”, which is expected to be a list of filenames). Added or updated keys are “filename”, “img”, “img\_shape”, “ori\_shape” (same as *img\_shape*), “pad\_shape” (same as *img\_shape*), “scale\_factor” (1.0) and “img\_norm\_cfg” (means=0 and stds=1).

#### Parameters

- **to\_float32** (*bool*) – Whether to convert the loaded image to a float32 numpy array. If set to False, the loaded image is an uint8 array. Defaults to False.
- **color\_type** (*str*) – The flag argument for `mmcv.imfrombytes()`. Defaults to ‘color’.
- **file\_client\_args** (*dict*) – Arguments to instantiate a FileClient. See `mmcv.fileio.FileClient` for details. Defaults to `dict(backend='disk')`.

**\_\_init\_\_** (*to\_float32=False, color\_type='unchanged', file\_client\_args={'backend': 'disk'}*)

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.detection.pipelines.mm_transforms.LoadAnnotations(with_bbox=True,
 with_label=True,
 with_mask=False,
 with_seg=False,
 poly2mask=True,
 file_client_args={'backend':
 'disk'})
```

Bases: object

Load multiple types of annotations.

#### Parameters

- **with\_bbox** (*bool*) – Whether to parse and load the bbox annotation. Default: True.
- **with\_label** (*bool*) – Whether to parse and load the label annotation. Default: True.
- **with\_mask** (*bool*) – Whether to parse and load the mask annotation. Default: False.
- **with\_seg** (*bool*) – Whether to parse and load the semantic segmentation annotation. Default: False.
- **poly2mask** (*bool*) – Whether to convert the instance masks from polygons to bitmaps. Default: True.
- **file\_client\_args** (*dict*) – Arguments to instantiate a FileClient. See `mmcv.fileio.FileClient` for details. Defaults to `dict(backend='disk')`.

```
__init__(with_bbox=True, with_label=True, with_mask=False, with_seg=False, poly2mask=True,
 file_client_args={'backend': 'disk'})
```

Initialize self. See `help(type(self))` for accurate signature.

#### process\_polygons(*polygons*)

Convert polygons to list of ndarray and filter invalid polygons.

**Parameters** **polygons** (*list[list]*) – Polygons of one instance.

**Returns** Processed polygons.

**Return type** `list[numpy.ndarray]`

```
class easycv.datasets.detection.pipelines.mm_transforms.MMMultiScaleFlipAug(transforms,
 img_scale=None,
 scale_factor=None,
 flip=False,
 flip_direction='horizontal')
```

Bases: object

Test-time augmentation with multiple scales and flipping.

An example configuration is as followed:

```
img_scale=[(1333, 400), (1333, 800)],
flip=True,
transforms=[
 dict(type='Resize', keep_ratio=True),
 dict(type='RandomFlip'),
 dict(type='Normalize', **img_norm_cfg),
 dict(type='Pad', size_divisor=32),
 dict(type='ImageToTensor', keys=['img']),
 dict(type='Collect', keys=['img']),
]
```

After MultiScaleFLipAug with above configuration, the results are wrapped into lists of the same length as followed:

```
dict(
 img=[...],
 img_shape=[...],
 scale=[(1333, 400), (1333, 400), (1333, 800), (1333, 800)]
 flip=[False, True, False, True]
 ...
)
```

### Parameters

- **transforms** (*list[dict]*) – Transforms to apply in each augmentation.
- **img\_scale** (*tuple | list[tuple] | None*) – Images scales for resizing.
- **scale\_factor** (*float | list[float] | None*) – Scale factors for resizing.
- **flip** (*bool*) – Whether apply flip augmentation. Default: False.
- **flip\_direction** (*str | list[str]*) – Flip augmentation directions, options are “horizontal”, “vertical” and “diagonal”. If flip\_direction is a list, multiple flip augmentations will be applied. It has no effect when flip == False. Default: “horizontal”.

**\_\_init\_\_** (*transforms, img\_scale=None, scale\_factor=None, flip=False, flip\_direction='horizontal'*)  
Initialize self. See help(type(self)) for accurate signature.

## Submodules

### easycv.datasets.detection.mix module

```
class easycv.datasets.detection.mix.DetImagesMixDataset(data_source, pipeline,
 dynamic_scale=None,
 skip_type_keys=None, profiling=False,
 classes=None, yolo_format=True,
 label_padding=True)
```

Bases: `Generic[torch.utils.data.dataset.T_co]`

A wrapper of multiple images mixed dataset.

Suitable for training on multiple images mixed data augmentation like mosaic and mixup. For the augmentation pipeline of mixed image data, the *get\_indexes* method needs to be provided to obtain the image indexes, and you can set *skip\_flags* to change the pipeline running process. At the same time, we provide the *dynamic\_scale* parameter to dynamically change the output image size.

output boxes format: cx, cy, w, h

### Parameters

- **data\_source** (`DetSourceCoco`) – The dataset to be mixed.
- **pipeline** (*Sequence[dict]*) – Sequence of transform object or config dict to be composed.
- **dynamic\_scale** (*tuple[int], optional*) – The image scale can be changed dynamically. Default to None.

- **skip\_type\_keys** (*list[str]*, *optional*) – Sequence of type string to be skip pipeline. Default to None.
- **label\_padding** – out labeling padding [N, 120, 5]

**\_\_init\_\_** (*data\_source*, *pipeline*, *dynamic\_scale=None*, *skip\_type\_keys=None*, *profiling=False*, *classes=None*, *yolo\_format=True*, *label\_padding=True*)

Args: *data\_source*: Data\_source config dict *pipeline*: Pipeline config list *profiling*: If set True, will print pipeline time *classes*: A list of class names, used in evaluation for result and groundtruth visualization

**update\_skip\_type\_keys** (*skip\_type\_keys*)

Update skip\_type\_keys. It is called by an external hook.

**Parameters** **skip\_type\_keys** (*list[str]*, *optional*) – Sequence of type string to be skip pipeline.

**update\_dynamic\_scale** (*dynamic\_scale*)

Update dynamic\_scale. It is called by an external hook.

**Parameters** **dynamic\_scale** (*tuple[int]*) – The image scale can be changed dynamically.

**results2json** (*results*, *outfile\_prefix*)

Dump the detection results to a COCO style json file.

There are 3 types of results: proposals, bbox predictions, mask predictions, and they have different data types. This method will automatically recognize the type, and dump them to json files.

#### Parameters

- **results** (*list[list | tuple | ndarray]*) – Testing results of the dataset.
- **outfile\_prefix** (*str*) – The filename prefix of the json files. If the prefix is “somepath/xxx”, the json files will be named “somepath/xxx.bbox.json”, “somepath/xxx.segm.json”, “somepath/xxx.proposal.json”.

**Returns** *str*: Possible keys are “bbox”, “segm”, “proposal”, and values are corresponding file-names.

**Return type** *dict[str]*

**format\_results** (*results*, *jsonfile\_prefix=None*, *\*\*kwargs*)

Format the results to json (standard format for COCO evaluation).

#### Parameters

- **results** (*list[tuple | numpy.ndarray]*) – Testing results of the dataset.
- **jsonfile\_prefix** (*str | None*) – The prefix of json files. It includes the file path and the prefix of filename, e.g., “a/b/prefix”. If not specified, a temp file will be created. Default: None.

**Returns** (*result\_files*, *tmp\_dir*), *result\_files* is a dict containing the json filepaths, *tmp\_dir* is the temporal directory created for saving json files when *jsonfile\_prefix* is not specified.

**Return type** *tuple*

**easycv.datasets.detection.raw module**

**class** easycv.datasets.detection.raw.**DetDataset**(*data\_source*, *pipeline*, *profiling=False*, *classes=None*)  
 Bases: Generic[torch.utils.data.dataset.T\_co]

Dataset for Detection

**\_\_init\_\_**(*data\_source*, *pipeline*, *profiling=False*, *classes=None*)

**Parameters**

- **data\_source** – Data\_source config dict
- **pipeline** – Pipeline config list
- **profiling** – If set True, will print pipeline time
- **classes** – A list of class names, used in evaluation for result and groundtruth visualization

**evaluate**(*results*, *evaluators=None*, *logger=None*)

Evaluates the detection boxes. :param results: A dictionary containing

**detection\_boxes:** List of length number of test images. Float32 numpy array of shape [num\_boxes, 4] and format [ymin, xmin, ymax, xmax] in absolute image coordinates.

**detection\_scores:** List of length number of test images, detection scores for the boxes, float32 numpy array of shape [num\_boxes].

**detection\_classes:** List of length number of test images, integer numpy array of shape [num\_boxes] containing 1-indexed detection classes for the boxes.

**img metas:** List of length number of test images, dict of image meta info, containing file-name, img\_shape, origin\_img\_shape, scale\_factor and so on.

**Parameters evaluators** – evaluators to calculate metric with results and groundtruth\_dict

**visualize**(*results*, *vis\_num=10*, *score\_thr=0.3*, *\*\*kwargs*)

Visulaize the model output on validation data. :param results: A dictionary containing

**detection\_boxes:** List of length number of test images. Float32 numpy array of shape [num\_boxes, 4] and format [ymin, xmin, ymax, xmax] in absolute image coordinates.

**detection\_scores:** List of length number of test images, detection scores for the boxes, float32 numpy array of shape [num\_boxes].

**detection\_classes:** List of length number of test images, integer numpy array of shape [num\_boxes] containing 1-indexed detection classes for the boxes.

**img metas:** List of length number of test images, dict of image meta info, containing file-name, img\_shape, origin\_img\_shape, scale\_factor and so on.

**Parameters**

- **vis\_num** – number of images visualized
- **score\_thr** – The threshold to filter box, boxes with scores greater than score\_thr will be kept.

**Returns:** A dictionary containing images: Visulaized images. img\_metas: List of length number of test images,

dict of image meta info, containing filename, img\_shape, origin\_img\_shape, scale\_factor and so on.

### 12.1.3 easycv.datasets.loader package

**class** easycv.datasets.loader.**GroupSampler**(dataset, samples\_per\_gpu=1)

Bases: Generic[torch.utils.data.sampler.T\_co]

**\_\_init\_\_**(dataset, samples\_per\_gpu=1)

Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.loader.**DistributedGroupSampler**(dataset, samples\_per\_gpu=1,  
num\_replicas=None, rank=None)

Bases: Generic[torch.utils.data.sampler.T\_co]

Sampler that restricts data loading to a subset of the dataset.

It is especially useful in conjunction with `torch.nn.parallel.DistributedDataParallel`. In such case, each process can pass a `DistributedSampler` instance as a `DataLoader` sampler, and load a subset of the original dataset that is exclusive to it. .. note:

Dataset **is** assumed to be of constant size.

#### Parameters

- **dataset** – Dataset used for sampling.
- **num\_replicas** (*optional*) – Number of processes participating in distributed training.
- **rank** (*optional*) – Rank of the current process within num\_replicas.

**\_\_init\_\_**(dataset, samples\_per\_gpu=1, num\_replicas=None, rank=None)

Initialize self. See help(type(self)) for accurate signature.

**set\_epoch**(epoch)

**easycv.datasets.loader.build\_data\_loader**(dataset, imgs\_per\_gpu, workers\_per\_gpu, num\_gpus=1,  
dist=True, shuffle=True, replace=False, seed=None,  
reuse\_worker\_cache=False, odps\_config=None,  
persistent\_workers=False, \*\*kwargs)

Build PyTorch DataLoader.

In distributed training, each GPU/process has a dataloader. In non-distributed training, there is only one dataloader for all GPUs.

#### Parameters

- **dataset** (*Dataset*) – A PyTorch dataset.
- **imgs\_per\_gpu** (*int*) – Number of images on each GPU, i.e., batch size of each GPU.
- **workers\_per\_gpu** (*int*) – How many subprocesses to use for data loading for each GPU.
- **num\_gpus** (*int*) – Number of GPUs. Only used in non-distributed training.
- **dist** (*bool*) – Distributed training/test or not. Default: True.
- **shuffle** (*bool*) – Whether to shuffle the data at every epoch. Default: True.
- **replace** (*bool*) – Replace or not in random shuffle. It works on when shuffle is True.

- **reuse\_worker\_cache** (*bool*) – If set true, will reuse worker process so that cached data in worker process can be reused.
- **persistent\_workers** (*bool*) – After pytorch1.7, could use persistent\_workers=True to avoid reconstruct dataloader before each epoch, speed up before epoch
- **kwargs** – any keyword argument to be used to initialize DataLoader

**Returns** A PyTorch dataloader.

**Return type** DataLoader

```
class easycv.datasets.loader.DistributedGivenIterationSampler(dataset, total_iter, batch_size,
 num_replicas=None, rank=None,
 last_iter=-1)
```

Bases: Generic[torch.utils.data.sampler.T\_co]

```
__init__(dataset, total_iter, batch_size, num_replicas=None, rank=None, last_iter=-1)
 Initialize self. See help(type(self)) for accurate signature.
```

```
set_uniform_indices(labels, num_classes)
```

```
gen_new_list()
```

```
set_epoch(epoch)
```

## Submodules

### easycv.datasets.loader.build\_loader module

```
easycv.datasets.loader.build_loader.build_dataloader(dataset, imgs_per_gpu, workers_per_gpu,
 num_gpus=1, dist=True, shuffle=True,
 replace=False, seed=None,
 reuse_worker_cache=False,
 odps_config=None, persistent_workers=False,
 **kwargs)
```

Build PyTorch DataLoader.

In distributed training, each GPU/process has a dataloader. In non-distributed training, there is only one dataloader for all GPUs.

#### Parameters

- **dataset** (*Dataset*) – A PyTorch dataset.
- **imgs\_per\_gpu** (*int*) – Number of images on each GPU, i.e., batch size of each GPU.
- **workers\_per\_gpu** (*int*) – How many subprocesses to use for data loading for each GPU.
- **num\_gpus** (*int*) – Number of GPUs. Only used in non-distributed training.
- **dist** (*bool*) – Distributed training/test or not. Default: True.
- **shuffle** (*bool*) – Whether to shuffle the data at every epoch. Default: True.
- **replace** (*bool*) – Replace or not in random shuffle. It works on when shuffle is True.
- **reuse\_worker\_cache** (*bool*) – If set true, will reuse worker process so that cached data in worker process can be reused.
- **persistent\_workers** (*bool*) – After pytorch1.7, could use persistent\_workers=True to avoid reconstruct dataloader before each epoch, speed up before epoch
- **kwargs** – any keyword argument to be used to initialize DataLoader

**Returns** A PyTorch dataloader.

**Return type** DataLoader

`easycv.datasets.loader.build_loader.worker_init_fn(worker_id, seed=None, odps_config=None)`

**class** `easycv.datasets.loader.build_loader.InfiniteDataLoader(*args, **kwargs)`

Bases: `Generic[torch.utils.data.dataloader.T_co]`

Dataloader that reuses workers. <https://github.com/pytorch/pytorch/issues/15849>

Uses same syntax as vanilla DataLoader.

**\_\_init\_\_**(\*args, \*\*kwargs)

Initialize self. See help(type(self)) for accurate signature.

**dataset:** `torch.utils.data.dataset.Dataset[torch.utils.data.dataloader.T_co]`

**batch\_size:** `Optional[int]`

**num\_workers:** `int`

**pin\_memory:** `bool`

**drop\_last:** `bool`

**timeout:** `float`

**sampler:** `Union[torch.utils.data.sampler.Sampler, Iterable]`

**prefetch\_factor:** `int`

### `easycv.datasets.loader.sampler` module

**class** `easycv.datasets.loader.sampler.DistributedMPSampler(dataset, num_replicas=None, rank=None, shuffle=True, split_huge_listfile_byrank=False)`

Bases: `torch.utils.data.sampler.Sampler[torch.utils.data.distributed.T_co]`

**\_\_init\_\_**(dataset, num\_replicas=None, rank=None, shuffle=True, split\_huge\_listfile\_byrank=False)

A Distribute sampler which support sample m instance from one class once for classification dataset

**dataset:** pytorch dataset object **num\_replicas** (optional): Number of processes participating in distributed training.

**rank** (optional): Rank of the current process within num\_replicas. **shuffle** (optional): If true (default), sampler will shuffle the indices **split\_huge\_listfile\_byrank**: if split, return all indice for each rank, because list for each rank has been

split before build dataset in dist training

**generate\_indice()**

**get\_label\_dict()**

**calculate\_this\_label\_list()**

**class** `easycv.datasets.loader.sampler.DistributedSampler(dataset, num_replicas=None, rank=None, shuffle=True, replace=False, split_huge_listfile_byrank=False)`

Bases: `torch.utils.data.sampler.Sampler[torch.utils.data.distributed.T_co]`



```
__init__(dataset, num_replicas=None, rank=None, shuffle=True, replace=False,
 split_huge_listfile_byrank=False)
```

A Distribute sampler which support sample m instance from one class once for classification dataset

#### Parameters

- **dataset** – pytorch dataset object
- **num\_replicas** (*optional*) – Number of processes participating in distributed training.
- **rank** (*optional*) – Rank of the current process within num\_replicas.
- **shuffle** (*optional*) – If true (default), sampler will shuffle the indices
- **split\_huge\_listfile\_byrank** – if split, return all indice for each rank, because list for each rank has been split before build dataset in dist training

```
generate_new_list()
```

```
set_uniform_indices(labels, num_classes)
```

```
class easycv.datasets.loader.sampler.GroupSampler(dataset, samples_per_gpu=1)
```

Bases: Generic[torch.utils.data.sampler.T\_co]

```
__init__(dataset, samples_per_gpu=1)
```

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.loader.sampler.DistributedGroupSampler(dataset, samples_per_gpu=1,
 num_replicas=None, rank=None)
```

Bases: Generic[torch.utils.data.sampler.T\_co]

Sampler that restricts data loading to a subset of the dataset.

It is especially useful in conjunction with `torch.nn.parallel.DistributedDataParallel`. In such case, each process can pass a `DistributedSampler` instance as a `DataLoader` sampler, and load a subset of the original dataset that is exclusive to it. .. note:

Dataset **is** assumed to be of constant size.

#### Parameters

- **dataset** – Dataset used for sampling.
- **num\_replicas** (*optional*) – Number of processes participating in distributed training.
- **rank** (*optional*) – Rank of the current process within num\_replicas.

```
__init__(dataset, samples_per_gpu=1, num_replicas=None, rank=None)
```

Initialize self. See help(type(self)) for accurate signature.

```
set_epoch(epoch)
```

```
class easycv.datasets.loader.sampler.DistributedGivenIterationSampler(dataset, total_iter,
 batch_size,
 num_replicas=None,
 rank=None, last_iter=-1)
```

Bases: Generic[torch.utils.data.sampler.T\_co]

```
__init__(dataset, total_iter, batch_size, num_replicas=None, rank=None, last_iter=-1)
```

Initialize self. See help(type(self)) for accurate signature.

```
set_uniform_indices(labels, num_classes)
```

```
gen_new_list()
set_epoch(epoch)
```

### 12.1.4 easycv.datasets.pose package

```
class easycv.datasets.pose.PoseTopDownDataset(data_source, pipeline, profiling=False)
```

Bases: `Generic[torch.utils.data.dataset.T_co]`

PoseTopDownDataset dataset for top-down pose estimation. The dataset loads raw features and apply specified transforms to return a dict containing the image tensors and other information.

#### Parameters

- **data\_source** – Data\_source config dict
- **pipeline** – Pipeline config list
- **profiling** – If set True, will print pipeline time

```
__init__(data_source, pipeline, profiling=False)
```

Initialize self. See help(type(self)) for accurate signature.

```
evaluate(outputs, evaluators, **kwargs)
```

#### Subpackages

##### easycv.datasets.pose.data\_sources package

```
class easycv.datasets.pose.data_sources.PoseTopDownSourceCoco(ann_file, img_prefix, data_cfg,
 dataset_info=None,
 test_mode=False)
```

Bases: `easycv.datasets.pose.data_sources.top_down.PoseTopDownSource`

CocoSource for top-down pose estimation.

Microsoft COCO: Common Objects in Context' ECCV'2014 More details can be found in the `paper` .

The source loads raw features to build a data meta object containing the image info, annotation info and others.

COCO keypoint indexes:

```
0: 'nose',
1: 'left_eye',
2: 'right_eye',
3: 'left_ear',
4: 'right_ear',
5: 'left_shoulder',
6: 'right_shoulder',
7: 'left_elbow',
8: 'right_elbow',
9: 'left_wrist',
10: 'right_wrist',
11: 'left_hip',
12: 'right_hip',
13: 'left_knee',
14: 'right_knee',
```

(continues on next page)

(continued from previous page)

```

15: 'left_ankle',
16: 'right_ankle'

```

**Parameters**

- **ann\_file** (*str*) – Path to the annotation file.
- **img\_prefix** (*str*) – Path to a directory where images are held. Default: None.
- **data\_cfg** (*dict*) – config
- **dataset\_info** ([DatasetInfo](#)) – A class containing all dataset info.
- **test\_mode** (*bool*) – Store True when building test or
- **dataset. Default** (*validation*) – False.

```
__init__(ann_file, img_prefix, data_cfg, dataset_info=None, test_mode=False)
```

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.pose.data_sources.PoseTopDownSource(ann_file, img_prefix, data_cfg,
 dataset_info, coco_style=True,
 test_mode=False)
```

Bases: object

Class for keypoint 2D top-down pose estimation with single-view RGB image as the data source.

**Parameters**

- **ann\_file** (*str*) – Path to the annotation file.
- **img\_prefix** (*str*) – Path to a directory where images are held. Default: None.
- **data\_cfg** (*dict*) – config
- **dataset\_info** ([DatasetInfo](#)) – A class containing all dataset info.
- **coco\_style** (*bool*) – Whether the annotation json is coco-style. Default: True
- **test\_mode** (*bool*) – Store True when building test or validation dataset. Default: False.

```
__init__(ann_file, img_prefix, data_cfg, dataset_info, coco_style=True, test_mode=False)
```

Initialize self. See help(type(self)) for accurate signature.

```
load_image(image_file)
```

```
get_length()
```

Get the size of the dataset.

```
get_sample(idx)
```

## Submodules

### easycv.datasets.pose.data\_sources.coco module

```
class easycv.datasets.pose.data_sources.coco.PoseTopDownSourceCoco(ann_file, img_prefix,
 data_cfg,
 dataset_info=None,
 test_mode=False)
```

Bases: `easycv.datasets.pose.data_sources.top_down.PoseTopDownSource`

CocoSource for top-down pose estimation.

Microsoft COCO: Common Objects in Context' ECCV'2014 More details can be found in the [paper](#) .

The source loads raw features to build a data meta object containing the image info, annotation info and others.

COCO keypoint indexes:

```
0: 'nose',
1: 'left_eye',
2: 'right_eye',
3: 'left_ear',
4: 'right_ear',
5: 'left_shoulder',
6: 'right_shoulder',
7: 'left_elbow',
8: 'right_elbow',
9: 'left_wrist',
10: 'right_wrist',
11: 'left_hip',
12: 'right_hip',
13: 'left_knee',
14: 'right_knee',
15: 'left_ankle',
16: 'right_ankle'
```

#### Parameters

- **ann\_file** (*str*) – Path to the annotation file.
- **img\_prefix** (*str*) – Path to a directory where images are held. Default: None.
- **data\_cfg** (*dict*) – config
- **dataset\_info** (`DatasetInfo`) – A class containing all dataset info.
- **test\_mode** (*bool*) – Store True when building test or
- **dataset. Default** (*validation*) – False.

```
__init__(ann_file, img_prefix, data_cfg, dataset_info=None, test_mode=False)
Initialize self. See help(type(self)) for accurate signature.
```

**easycv.datasets.pose.data\_sources.top\_down module**

**class** easycv.datasets.pose.data\_sources.top\_down.**DatasetInfo**(*dataset\_info*)

Bases: object

**\_\_init\_\_**(*dataset\_info*)

Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.pose.data\_sources.top\_down.**PoseTopDownSource**(*ann\_file, img\_prefix, data\_cfg, dataset\_info, coco\_style=True, test\_mode=False*)

Bases: object

Class for keypoint 2D top-down pose estimation with single-view RGB image as the data source.

**Parameters**

- **ann\_file** (*str*) – Path to the annotation file.
- **img\_prefix** (*str*) – Path to a directory where images are held. Default: None.
- **data\_cfg** (*dict*) – config
- **dataset\_info** (**DatasetInfo**) – A class containing all dataset info.
- **coco\_style** (*bool*) – Whether the annotation json is coco-style. Default: True
- **test\_mode** (*bool*) – Store True when building test or validation dataset. Default: False.

**\_\_init\_\_**(*ann\_file, img\_prefix, data\_cfg, dataset\_info, coco\_style=True, test\_mode=False*)

Initialize self. See help(type(self)) for accurate signature.

**load\_image**(*image\_file*)

**get\_length**()

Get the size of the dataset.

**get\_sample**(*idx*)

**easycv.datasets.pose.pipelines package**

**class** easycv.datasets.pose.pipelines.**PoseCollect**(*keys, meta\_keys, meta\_name='img metas'*)

Bases: object

Collect data from the loader relevant to the specific task.

This keeps the items in *keys* as it is, and collect items in *meta\_keys* into a meta item called *meta\_name*. This is usually the last stage of the data loader pipeline. For example, when *keys*='imgs', *meta\_keys*=('filename', 'label', 'original\_shape'), *meta\_name*='img metas', the results will be a dict with keys 'imgs' and 'img metas', where 'img metas' is a DataContainer of another dict with keys 'filename', 'label', 'original\_shape'.

**Parameters**

- **keys** (*Sequence[str/tuple]*) – Required keys to be collected. If a tuple (key, key\_new) is given as an element, the item retrieved by key will be renamed as key\_new in collected data.
- **meta\_name** (*str*) – The name of the key that contains meta information. This key is always populated. Default: "img metas".
- **meta\_keys** (*Sequence[str/tuple]*) – Keys that are collected under meta\_name. The contents of the *meta\_name* dictionary depends on *meta\_keys*.

**\_\_init\_\_**(*keys, meta\_keys, meta\_name='img\_metas'*)  
Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.pose.pipelines.**TopDownRandomFlip**(*flip\_prob=0.5*)

Bases: object

Data augmentation with random image flip.

Required keys: 'img', 'joints\_3d', 'joints\_3d\_visible', 'center' and 'ann\_info'. Modifies key: 'img', 'joints\_3d', 'joints\_3d\_visible', 'center' and 'flipped'.

#### Parameters

- **flip** (*bool*) – Option to perform random flip.
- **flip\_prob** (*float*) – Probability of flip.

**\_\_init\_\_**(*flip\_prob=0.5*)  
Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.pose.pipelines.**TopDownHalfBodyTransform**(*num\_joints\_half\_body=8, prob\_half\_body=0.3*)

Bases: object

Data augmentation with half-body transform. Keep only the upper body or the lower body at random.

Required keys: 'joints\_3d', 'joints\_3d\_visible', and 'ann\_info'. Modifies key: 'scale' and 'center'.

#### Parameters

- **num\_joints\_half\_body** (*int*) – Threshold of performing half-body transform. If the body has fewer number of joints ( $< \text{num\_joints\_half\_body}$ ), ignore this step.
- **prob\_half\_body** (*float*) – Probability of half-body transform.

**\_\_init\_\_**(*num\_joints\_half\_body=8, prob\_half\_body=0.3*)  
Initialize self. See help(type(self)) for accurate signature.

**static half\_body\_transform**(*cfg, joints\_3d, joints\_3d\_visible*)  
Get center&scale for half-body transform.

**class** easycv.datasets.pose.pipelines.**TopDownGetRandomScaleRotation**(*rot\_factor=40, scale\_factor=0.5, rot\_prob=0.6*)

Bases: object

Data augmentation with random scaling & rotating.

Required key: 'scale'. Modifies key: 'scale' and 'rotation'.

#### Parameters

- **rot\_factor** (*int*) – Rotating to  $[-2*\text{rot\_factor}, 2*\text{rot\_factor}]$ .
- **scale\_factor** (*float*) – Scaling to  $[1-\text{scale\_factor}, 1+\text{scale\_factor}]$ .
- **rot\_prob** (*float*) – Probability of random rotation.

**\_\_init\_\_**(*rot\_factor=40, scale\_factor=0.5, rot\_prob=0.6*)  
Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.pose.pipelines.**TopDownAffine**(*use\_udp=False*)

Bases: object

Affine transform the image to make input.

Required keys: 'img', 'joints\_3d', 'joints\_3d\_visible', 'ann\_info', 'scale', 'rotation' and 'center'. Modified keys: 'img', 'joints\_3d', and 'joints\_3d\_visible'.

**Parameters** **use\_udp** (*bool*) – To use unbiased data processing. Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020).

**\_\_init\_\_** (*use\_udp=False*)

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.pose.pipelines.TopDownGenerateTarget(sigma=2, kernel=(11, 11),
 valid_radius_factor=0.0546875,
 target_type='GaussianHeatmap',
 encoding='MSRA',
 unbiased_encoding=False)
```

Bases: object

Generate the target heatmap.

Required keys: 'joints\_3d', 'joints\_3d\_visible', 'ann\_info'. Modified keys: 'target', and 'target\_weight'.

#### Parameters

- **sigma** – Sigma of heatmap gaussian for 'MSRA' approach.
- **kernel** – Kernel of heatmap gaussian for 'Megvii' approach.
- **encoding** (*str*) – Approach to generate target heatmaps. Currently supported approaches: 'MSRA', 'Megvii', 'UDP'. Default: 'MSRA'
- **unbiased\_encoding** (*bool*) – Option to use unbiased encoding methods. Paper ref: Zhang et al. Distribution-Aware Coordinate Representation for Human Pose Estimation (CVPR 2020).
- **keypoint\_pose\_distance** – Keypoint pose distance for UDP. Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020).
- **target\_type** (*str*) – supported targets: 'GaussianHeatmap', 'CombinedTarget'. Default: 'GaussianHeatmap' CombinedTarget: The combination of classification target (response map) and regression target (offset map). Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020).

**\_\_init\_\_** (*sigma=2, kernel=(11, 11), valid\_radius\_factor=0.0546875, target\_type='GaussianHeatmap', encoding='MSRA', unbiased\_encoding=False*)

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.pose.pipelines.TopDownGenerateTargetRegression
```

Bases: object

Generate the target regression vector (coordinates).

Required keys: 'joints\_3d', 'joints\_3d\_visible', 'ann\_info'. Modified keys: 'target', and 'target\_weight'.

**\_\_init\_\_** ()

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.pose.pipelines.TopDownRandomTranslation(trans_factor=0.15,
 trans_prob=1.0)
```

Bases: object

Data augmentation with random translation.

Required key: 'scale' and 'center'. Modifies key: 'center'.

## Notes

bbox height: H bbox width: W

### Parameters

- **trans\_factor** (*float*) – Translating center to  $[-\text{trans\_factor}, \text{trans\_factor}] * [W, H] + \text{center}$ .
- **trans\_prob** (*float*) – Probability of random translation.

**\_\_init\_\_**(*trans\_factor=0.15, trans\_prob=1.0*)

Initialize self. See help(type(self)) for accurate signature.

## Submodules

### easycv.datasets.pose.pipelines.transforms module

**class** easycv.datasets.pose.pipelines.transforms.**PoseCollect**(*keys, meta\_keys, meta\_name='img\_metas'*)

Bases: object

Collect data from the loader relevant to the specific task.

This keeps the items in *keys* as it is, and collect items in *meta\_keys* into a meta item called *meta\_name*. This is usually the last stage of the data loader pipeline. For example, when *keys*='imgs', *meta\_keys*=('filename', 'label', 'original\_shape'), *meta\_name*='img\_metas', the results will be a dict with keys 'imgs' and 'img\_metas', where 'img\_metas' is a DataContainer of another dict with keys 'filename', 'label', 'original\_shape'.

### Parameters

- **keys** (*Sequence[str/tuple]*) – Required keys to be collected. If a tuple (key, key\_new) is given as an element, the item retrieved by key will be renamed as key\_new in collected data.
- **meta\_name** (*str*) – The name of the key that contains meta information. This key is always populated. Default: "img\_metas".
- **meta\_keys** (*Sequence[str/tuple]*) – Keys that are collected under meta\_name. The contents of the *meta\_name* dictionary depends on *meta\_keys*.

**\_\_init\_\_**(*keys, meta\_keys, meta\_name='img\_metas'*)

Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.pose.pipelines.transforms.**TopDownRandomFlip**(*flip\_prob=0.5*)

Bases: object

Data augmentation with random image flip.

Required keys: 'img', 'joints\_3d', 'joints\_3d\_visible', 'center' and 'ann\_info'. Modifies key: 'img', 'joints\_3d', 'joints\_3d\_visible', 'center' and 'flipped'.

### Parameters

- **flip** (*bool*) – Option to perform random flip.
- **flip\_prob** (*float*) – Probability of flip.

**\_\_init\_\_**(*flip\_prob=0.5*)

Initialize self. See help(type(self)) for accurate signature.



```
class easycv.datasets.pose.pipelines.transforms.TopDownHalfBodyTransform(num_joints_half_body=8,
 prob_half_body=0.3)
```

Bases: object

Data augmentation with half-body transform. Keep only the upper body or the lower body at random.

Required keys: 'joints\_3d', 'joints\_3d\_visible', and 'ann\_info'. Modifies key: 'scale' and 'center'.

#### Parameters

- **num\_joints\_half\_body** (*int*) – Threshold of performing half-body transform. If the body has fewer number of joints ( $< \text{num\_joints\_half\_body}$ ), ignore this step.
- **prob\_half\_body** (*float*) – Probability of half-body transform.

```
__init__(num_joints_half_body=8, prob_half_body=0.3)
 Initialize self. See help(type(self)) for accurate signature.
```

```
static half_body_transform(cfg, joints_3d, joints_3d_visible)
 Get center&scale for half-body transform.
```

```
class easycv.datasets.pose.pipelines.transforms.TopDownGetRandomScaleRotation(rot_factor=40,
 scale_factor=0.5,
 rot_prob=0.6)
```

Bases: object

Data augmentation with random scaling & rotating.

Required key: 'scale'. Modifies key: 'scale' and 'rotation'.

#### Parameters

- **rot\_factor** (*int*) – Rotating to  $[-2*\text{rot\_factor}, 2*\text{rot\_factor}]$ .
- **scale\_factor** (*float*) – Scaling to  $[1-\text{scale\_factor}, 1+\text{scale\_factor}]$ .
- **rot\_prob** (*float*) – Probability of random rotation.

```
__init__(rot_factor=40, scale_factor=0.5, rot_prob=0.6)
 Initialize self. See help(type(self)) for accurate signature.
```

```
class easycv.datasets.pose.pipelines.transforms.TopDownAffine(use_udp=False)
```

Bases: object

Affine transform the image to make input.

Required keys: 'img', 'joints\_3d', 'joints\_3d\_visible', 'ann\_info', 'scale', 'rotation' and 'center'. Modified keys: 'img', 'joints\_3d', and 'joints\_3d\_visible'.

**Parameters** **use\_udp** (*bool*) – To use unbiased data processing. Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020).

```
__init__(use_udp=False)
 Initialize self. See help(type(self)) for accurate signature.
```

```
class easycv.datasets.pose.pipelines.transforms.TopDownGenerateTarget(sigma=2, kernel=(11,
 11),
 valid_radius_factor=0.0546875,
 tar-
 get_type='GaussianHeatmap',
 encoding='MSRA', unbi-
 ased_encoding=False)
```

Bases: object

Generate the target heatmap.

Required keys: 'joints\_3d', 'joints\_3d\_visible', 'ann\_info'. Modified keys: 'target', and 'target\_weight'.

#### Parameters

- **sigma** – Sigma of heatmap gaussian for 'MSRA' approach.
- **kernel** – Kernel of heatmap gaussian for 'Megvii' approach.
- **encoding** (*str*) – Approach to generate target heatmaps. Currently supported approaches: 'MSRA', 'Megvii', 'UDP'. Default: 'MSRA'
- **unbiased\_encoding** (*bool*) – Option to use unbiased encoding methods. Paper ref: Zhang et al. Distribution-Aware Coordinate Representation for Human Pose Estimation (CVPR 2020).
- **keypoint\_pose\_distance** – Keypoint pose distance for UDP. Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020).
- **target\_type** (*str*) – supported targets: 'GaussianHeatmap', 'CombinedTarget'. Default: 'GaussianHeatmap' CombinedTarget: The combination of classification target (response map) and regression target (offset map). Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020).

**\_\_init\_\_** (*sigma=2, kernel=(11, 11), valid\_radius\_factor=0.0546875, target\_type='GaussianHeatmap', encoding='MSRA', unbiased\_encoding=False*)

Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.pose.pipelines.transforms.**TopDownGenerateTargetRegression**

Bases: object

Generate the target regression vector (coordinates).

Required keys: 'joints\_3d', 'joints\_3d\_visible', 'ann\_info'. Modified keys: 'target', and 'target\_weight'.

**\_\_init\_\_** ()

Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.pose.pipelines.transforms.**TopDownRandomTranslation** (*trans\_factor=0.15, trans\_prob=1.0*)

Bases: object

Data augmentation with random translation.

Required key: 'scale' and 'center'. Modifies key: 'center'.

#### Notes

bbox height: H bbox width: W

#### Parameters

- **trans\_factor** (*float*) – Translating center to  $[-\text{trans\_factor}, \text{trans\_factor}] * [W, H] + \text{center}$ .
- **trans\_prob** (*float*) – Probability of random translation.

**\_\_init\_\_** (*trans\_factor=0.15, trans\_prob=1.0*)

Initialize self. See help(type(self)) for accurate signature.

## Submodules

### easycv.datasets.pose.top\_down module

**class** easycv.datasets.pose.top\_down.**PoseTopDownDataset**(*data\_source, pipeline, profiling=False*)  
 Bases: Generic[torch.utils.data.dataset.T\_co]

PoseTopDownDataset dataset for top-down pose estimation. The dataset loads raw features and apply specified transforms to return a dict containing the image tensors and other information.

#### Parameters

- **data\_source** – Data\_source config dict
- **pipeline** – Pipeline config list
- **profiling** – If set True, will print pipeline time

**\_\_init\_\_**(*data\_source, pipeline, profiling=False*)  
 Initialize self. See help(type(self)) for accurate signature.

**evaluate**(*outputs, evaluators, \*\*kwargs*)

## 12.1.5 easycv.datasets.selfsup package

### Subpackages

#### easycv.datasets.selfsup.data\_sources package

**class** easycv.datasets.selfsup.data\_sources.**SSLSourceImageList**(*list\_file, root="", max\_try=20*)  
 Bases: object

datasource for classification

#### Parameters

- **list\_file** – str / list(str), str means a input image list file path, this file contains records as *image\_path label* in list\_file list(str) means multi image list, each one contains some records as *image\_path label*
- **root** – str / list(str), root path for image\_path, each list\_file will need a root, if len(root) < len(list\_file), we will use root[-1] to fill root list.
- **max\_try** – int, max try numbers of reading image

**\_\_init\_\_**(*list\_file, root="", max\_try=20*)  
 Initialize self. See help(type(self)) for accurate signature.

**static parse\_list\_file**(*list\_file, root*)

**get\_length**()

**get\_sample**(*idx*)

**class** easycv.datasets.selfsup.data\_sources.**SSLSourceImageNetFeature**(*root\_path, training=True, data\_keyword='feat1', label\_keyword='label', dynamic\_load=True*)  
 Bases: object

```
__init__(root_path, training=True, data_keyword='feat1', label_keyword='label', dynamic_load=True)
 Initialize self. See help(type(self)) for accurate signature.

get_sample(idx)

get_length()
```

## Submodules

### easycv.datasets.selfsup.data\_sources.image\_list module

```
class easycv.datasets.selfsup.data_sources.image_list.SSLSourceImageList(list_file, root="",
 max_try=20)
```

Bases: object

datasource for classification

#### Parameters

- **list\_file** – str / list(str), str means a input image list file path, this file contains records as *image\_path label* in list\_file list(str) means multi image list, each one contains some records as *image\_path label*
- **root** – str / list(str), root path for image\_path, each list\_file will need a root, if len(root) < len(list\_file), we will use root[-1] to fill root list.
- **max\_try** – int, max try numbers of reading image

```
__init__(list_file, root="", max_try=20)
 Initialize self. See help(type(self)) for accurate signature.

static parse_list_file(list_file, root)

get_length()

get_sample(idx)
```

### easycv.datasets.selfsup.data\_sources.imagenet\_feature module

```
class easycv.datasets.selfsup.data_sources.imagenet_feature.SSLSourceImageNetFeature(root_path,
 train-
 ing=True,
 data_keyword='feat1',
 la-
 bel_keyword='label',
 dy-
 namic_load=True)
```

Bases: object

```
__init__(root_path, training=True, data_keyword='feat1', label_keyword='label', dynamic_load=True)
 Initialize self. See help(type(self)) for accurate signature.

get_sample(idx)

get_length()
```

**easycv.datasets.selfsup.pipelines package**

```
class easycv.datasets.selfsup.pipelines.RandomAppliedTrans(transforms, p=0.5)
 Bases: object

 Randomly applied transformations. :param transforms: List of transformations in dictionaries. :type transforms:
 List[Dict]

 __init__(transforms, p=0.5)
 Initialize self. See help(type(self)) for accurate signature.

class easycv.datasets.selfsup.pipelines.Lighting
 Bases: object

 Lighting noise(AlexNet - style PCA - based noise)

 __init__()
 Initialize self. See help(type(self)) for accurate signature.

class easycv.datasets.selfsup.pipelines.Solarization(threshold=128)
 Bases: object

 __init__(threshold=128)
 Initialize self. See help(type(self)) for accurate signature.
```

**Submodules****easycv.datasets.selfsup.pipelines.transforms module**

```
class easycv.datasets.selfsup.pipelines.transforms.MAEFtAugment(input_size=None,
 color_jitter=None,
 auto_augment=None,
 interpolation=None,
 re_prob=None, re_mode=None,
 re_count=None, mean=None,
 std=None, is_train=True)

 Bases: object

 RandAugment data augmentation method based on “RandAugment: Practical automated data augmentation with a reduced search space”. This code is borrowed from <https://github.com/pengzhiliang/MAE-pytorch> :param
 input_size: images input size :type input_size: int :param color_jitter: Color jitter factor :type color_jitter: float
 :param auto_augment: Use AutoAugment policy :param interpolation: Training interpolation :param re_prob:
 Random erase prob :param re_mode: Random erase mode :param re_count: Random erase count :param mean:
 mean used for normalization :param std: std used for normalization :param is_train: If True use all augmentation
 strategy

 __init__(input_size=None, color_jitter=None, auto_augment=None, interpolation=None, re_prob=None,
 re_mode=None, re_count=None, mean=None, std=None, is_train=True)
 Initialize self. See help(type(self)) for accurate signature.

class easycv.datasets.selfsup.pipelines.transforms.RandomAppliedTrans(transforms, p=0.5)
 Bases: object

 Randomly applied transformations. :param transforms: List of transformations in dictionaries. :type transforms:
 List[Dict]

 __init__(transforms, p=0.5)
 Initialize self. See help(type(self)) for accurate signature.
```

```
class easycv.datasets.selfsup.pipelines.transforms.Lighting
```

Bases: object

Lighting noise(AlexNet - style PCA - based noise)

```
__init__()
```

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.selfsup.pipelines.transforms.Solarization(threshold=128)
```

Bases: object

```
__init__(threshold=128)
```

Initialize self. See help(type(self)) for accurate signature.

### 12.1.6 easycv.datasets.shared package

```
class easycv.datasets.shared.ConcatDataset(datasets)
```

Bases: torch.utils.data.dataset.Dataset[torch.utils.data.dataset.T\_co]

A wrapper of concatenated dataset.

Same as torch.utils.data.dataset.ConcatDataset, but concat the group flag for image aspect ratio.

**Parameters** **datasets** (list[Dataset]) – A list of datasets.

```
__init__(datasets)
```

Initialize self. See help(type(self)) for accurate signature.

**datasets:** List[torch.utils.data.dataset.Dataset[torch.utils.data.dataset.T\_co]]

**cumulative\_sizes:** List[int]

```
class easycv.datasets.shared.RepeatDataset(dataset, times)
```

Bases: object

A wrapper of repeated dataset.

The length of repeated dataset will be *times* larger than the original dataset. This is useful when the data loading time is long but the dataset is small. Using RepeatDataset can reduce the data loading time between epochs.

**Parameters**

- **dataset** (Dataset) – The dataset to be repeated.
- **times** (int) – Repeat times.

```
__init__(dataset, times)
```

Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.shared.OdpsReader(table_name, selected_cols=[], excluded_cols=[],
 random_start=False, odps_io_config=None,
 image_col='url_image', image_type='url')
```

Bases: object

```
__init__(table_name, selected_cols=[], excluded_cols=[], random_start=False, odps_io_config=None,
 image_col='url_image', image_type='url')
```

Init odps reader and datasource set to load data from odps table

**Parameters**

- **table\_name** (str) – odps table to load
- **selected\_cols** (list(str)) – select column

- **excluded\_cols** (*list(str)*) – exclude column
- **random\_start** (*bool*) – random start for odps table
- **odps\_io\_config** (*dict*) – odps config contains access\_id, access\_key, endpoint
- **image\_col** (*list(str)*) – image column names
- **image\_type** (*list(str)*) – image column types support url/base64, must be same length with image type or 0

**Returns :** None

**get\_length()**

**reset\_reader**(*dataloader\_workid, dataloader\_worknum*)

**get\_sample**(*idx*)

**b64\_decode()**

**class** easycv.datasets.shared.**RawDataset**(*data\_source, pipeline*)

Bases: Generic[torch.utils.data.dataset.T\_co]

**\_\_init\_\_**(*data\_source, pipeline*)

Initialize self. See help(type(self)) for accurate signature.

**evaluate**(*scores, keyword, logger=None*)

**class** easycv.datasets.shared.**BaseDataset**(*data\_source, pipeline, profiling=False*)

Bases: Generic[torch.utils.data.dataset.T\_co]

Base Dataset

**\_\_init\_\_**(*data\_source, pipeline, profiling=False*)

Initialize self. See help(type(self)) for accurate signature.

**abstract evaluate**(*results, evaluators, logger=None, \*\*kwargs*)

**visualize**(*results, \*\*kwargs*)

Visualize the model output results on validation data. Returns: A dictionary

**If add image visualization, return dict containing** images: List of visualized images.  
img metas: List of length number of test images,

dict of image meta info, containing filename, img\_shape, origin\_img\_shape,  
scale\_factor and so on.

**class** easycv.datasets.shared.**MultiViewDataset**(*data\_source, num\_views, pipelines*)

Bases: Generic[torch.utils.data.dataset.T\_co]

The dataset outputs multiple views of an image. The number of views in the output dict depends on *num\_views*. The image can be processed by one pipeline or multiple pipelines. :param num\_views: The number of different views. :type num\_views: list :param pipelines: A list of pipelines. :type pipelines: list[list[dict]]

**\_\_init\_\_**(*data\_source, num\_views, pipelines*)

Initialize self. See help(type(self)) for accurate signature.

**evaluate**(*results, evaluators, logger=None*)

## Subpackages

### easycv.datasets.shared.data\_sources package

```
class easycv.datasets.shared.data_sources.ImageNpy(image_file, label_file=None,
 cache_root='data_cache/')
 Bases: object
 __init__(image_file, label_file=None, cache_root='data_cache/')
 image_file: (local or oss) image data saved in one .npy data [cv2.img, cv2.img,...] label_file: (local or oss)
 label data saved in one .npy data
 get_length()
 get_sample(idx)
class easycv.datasets.shared.data_sources.SourceConcat(data_source_list)
 Bases: object
 Concat multi data source config.
 __init__(data_source_list)
 Initialize self. See help(type(self)) for accurate signature.
 get_length()
 get_sample(idx)
 cumsum_length()
```

## Submodules

### easycv.datasets.shared.data\_sources.concat module

```
class easycv.datasets.shared.data_sources.concat.SourceConcat(data_source_list)
 Bases: object
 Concat multi data source config.
 __init__(data_source_list)
 Initialize self. See help(type(self)) for accurate signature.
 get_length()
 get_sample(idx)
 cumsum_length()
```

### easycv.datasets.shared.data\_sources.image\_npy module

```
class easycv.datasets.shared.data_sources.image_npy.ImageNpy(image_file, label_file=None,
 cache_root='data_cache/')
 Bases: object
 __init__(image_file, label_file=None, cache_root='data_cache/')
 image_file: (local or oss) image data saved in one .npy data [cv2.img, cv2.img,...] label_file: (local or oss)
 label data saved in one .npy data
 get_length()
```



`get_sample(idx)`

## easycv.datasets.shared.pipelines package

### Submodules

#### easycv.datasets.shared.pipelines.dali\_transforms module

**class** easycv.datasets.shared.pipelines.dali\_transforms.**DaliImageDecoder**(*device='mixed', output\_type=DALIImageType.RGB, \*\*kwargs*)

Bases: object

refer to: [https://docs.nvidia.com/deeplearning/dali/archives/dali\\_0250/user-guide/docs/supported\\_ops.html#nvidia.dali.ops.ImageDecoder](https://docs.nvidia.com/deeplearning/dali/archives/dali_0250/user-guide/docs/supported_ops.html#nvidia.dali.ops.ImageDecoder)

**\_\_init\_\_**(*device='mixed', output\_type=DALIImageType.RGB, \*\*kwargs*)  
Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.shared.pipelines.dali\_transforms.**DaliRandomResizedCrop**(*size, random\_area, device='gpu', \*\*kwargs*)

Bases: object

refer to: [https://docs.nvidia.com/deeplearning/dali/archives/dali\\_0250/user-guide/docs/supported\\_ops.html#nvidia.dali.ops.RandomResizedCrop](https://docs.nvidia.com/deeplearning/dali/archives/dali_0250/user-guide/docs/supported_ops.html#nvidia.dali.ops.RandomResizedCrop)

**\_\_init\_\_**(*size, random\_area, device='gpu', \*\*kwargs*)  
Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.shared.pipelines.dali\_transforms.**DaliResize**(*resize\_shorter, device='gpu', interp\_type=DALIInterpType.INTERP\_TRIANGULAR, \*\*kwargs*)

Bases: object

refer to: [https://docs.nvidia.com/deeplearning/dali/archives/dali\\_0250/user-guide/docs/supported\\_ops.html#nvidia.dali.ops.Resize](https://docs.nvidia.com/deeplearning/dali/archives/dali_0250/user-guide/docs/supported_ops.html#nvidia.dali.ops.Resize)

**\_\_init\_\_**(*resize\_shorter, device='gpu', interp\_type=DALIInterpType.INTERP\_TRIANGULAR, \*\*kwargs*)  
Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.shared.pipelines.dali\_transforms.**DaliColorTwist**(*prob, saturation, contrast, brightness, hue, device='gpu', center=1*)

Bases: object

refer to: [https://docs.nvidia.com/deeplearning/dali/archives/dali\\_0250/user-guide/docs/supported\\_ops.html#nvidia.dali.ops.ColorTwist](https://docs.nvidia.com/deeplearning/dali/archives/dali_0250/user-guide/docs/supported_ops.html#nvidia.dali.ops.ColorTwist)

**\_\_init\_\_**(*prob, saturation, contrast, brightness, hue, device='gpu', center=1*)  
Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.shared.pipelines.dali\_transforms.**DaliRandomGrayscale**(*prob, device='gpu'*)

Bases: object

refer to: [https://docs.nvidia.com/deeplearning/dali/archives/dali\\_0250/user-guide/docs/supported\\_ops.html#nvidia.dali.ops.Hsv](https://docs.nvidia.com/deeplearning/dali/archives/dali_0250/user-guide/docs/supported_ops.html#nvidia.dali.ops.Hsv) Create *RandomGrayscale* op with ops.Hsv. when saturation=0, it represents a grayscale image

**\_\_init\_\_**(*prob, device='gpu'*)  
Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.shared.pipelines.dali_transforms.DaliCropMirrorNormalize(crop, mean,
 std,
 prob=0.0,
 device='gpu',
 dtype=DALIDataType.FLOAT,
 out-
 put_layout='CHW',
 crop_pos_x=0.5,
 crop_pos_y=0.5,
 **kwargs)
```

Bases: object

refer to: [https://docs.nvidia.com/deeplearning/dali/archives/dali\\_0250/user-guide/docs/supported\\_ops.html#nvidia.dali.ops.CropMirrorNormalize](https://docs.nvidia.com/deeplearning/dali/archives/dali_0250/user-guide/docs/supported_ops.html#nvidia.dali.ops.CropMirrorNormalize)

**\_\_init\_\_**(*crop, mean, std, prob=0.0, device='gpu', dtype=DALIDataType.FLOAT, output\_layout='CHW',*  
 *crop\_pos\_x=0.5, crop\_pos\_y=0.5, \*\*kwargs*)  
Initialize self. See help(type(self)) for accurate signature.

## easycv.datasets.shared.pipelines.format module

easycv.datasets.shared.pipelines.format.to\_tensor(*data*)

Convert objects of various python types to torch.Tensor.

Supported types are: numpy.ndarray, torch.Tensor, Sequence, int and float.

**Parameters** *data* (*torch.Tensor* | *numpy.ndarray* | *Sequence* | *int* | *float*) – Data to be converted.

class easycv.datasets.shared.pipelines.format.ImageToTensor(*keys*)

Bases: object

Convert image to torch.Tensor by given keys.

The dimension order of input image is (H, W, C). The pipeline will convert it to (C, H, W). If only 2 dimension (H, W) is given, the output would be (1, H, W).

**Parameters** *keys* (*Sequence[str]*) – Key of images to be converted to Tensor.

**\_\_init\_\_**(*keys*)  
Initialize self. See help(type(self)) for accurate signature.

```
class easycv.datasets.shared.pipelines.format.Collect(keys, meta_keys=
 ('filename', 'ori_filename',
 'ori_img_shape', 'img_shape', 'scale_factor',
 'pad', 'flip', 'flip_direction', 'img_norm_cfg'))
```

Bases: object

Collect data from the loader relevant to the specific task.

This is usually the last stage of the data loader pipeline. Typically keys is set to some subset of “img”, “proposals”, “gt\_bboxes”, “gt\_bboxes\_ignore”, “gt\_labels”, and/or “gt\_masks”.

The “img\_meta” item is always populated. The contents of the “img\_meta” dictionary depends on “meta\_keys”. By default this includes:

- “img\_shape”: shape of the image input to the network as a tuple (h, w). Note that images may be zero padded on the bottom/right if the batch tensor is larger than this shape.
- “scale\_factor”: a float indicating the preprocessing scale
- “flip”: a boolean indicating if image flip transform was used
- “filename”: path to the image file
- “ori\_img\_shape”: original shape of the image as a tuple (h, w, c)
- “img\_norm\_cfg”: a dict of normalization information:
  - mean - per channel mean subtraction
  - std - per channel std divisor
  - to\_rgb - bool indicating if bgr was converted to rgb

#### Parameters

- **keys** (*Sequence[str]*) – Keys of results to be collected in data.
- **meta\_keys** (*Sequence[str], optional*) – Meta keys to be converted to mmcv. DataContainer and collected in `data[img_metas]`. Default: `((‘filename’, ‘ori_filename’, ‘ori_img_shape’, ‘img_shape’, ‘scale_factor’, ‘flip’, ‘flip_direction’, ‘img_norm_cfg’))`

**\_\_init\_\_**(*keys, meta\_keys=(‘filename’, ‘ori\_filename’, ‘ori\_img\_shape’, ‘img\_shape’, ‘scale\_factor’, ‘pad’, ‘flip’, ‘flip\_direction’, ‘img\_norm\_cfg’)*)

Initialize self. See help(type(self)) for accurate signature.

**class** easycv.datasets.shared.pipelines.format.DefaultFormatBundle

Bases: object

Default formatting bundle.

It simplifies the pipeline of formatting common fields, including “img”, “proposals”, “gt\_bboxes”, “gt\_labels”, “gt\_masks” and “gt\_semantic\_seg”. These fields are formatted as follows.

- img: (1)transpose, (2)to tensor, (3)to DataContainer (stack=True)
- proposals: (1)to tensor, (2)to DataContainer
- gt\_bboxes: (1)to tensor, (2)to DataContainer
- gt\_bboxes\_ignore: (1)to tensor, (2)to DataContainer
- gt\_labels: (1)to tensor, (2)to DataContainer
- gt\_masks: (1)to tensor, (2)to DataContainer (cpu\_only=True)
- gt\_semantic\_seg: (1)unsqueeze dim-0 (2)to tensor, (3)to DataContainer (stack=True)

**easycv.datasets.shared.pipelines.third\_transforms\_wrapper module**

`easycv.datasets.shared.pipelines.third_transforms_wrapper.is_child_of(obj, cls)`

`easycv.datasets.shared.pipelines.third_transforms_wrapper.get_args(obj)`

`easycv.datasets.shared.pipelines.third_transforms_wrapper.wrap_torchvision_transforms(transform_obj)`

**class** `easycv.datasets.shared.pipelines.third_transforms_wrapper.AutoAugment`(*policy: torchvision.transforms.autoaugment.AutoAugmentPolicy.IMAGENET: 'imagenet', interpolation: torchvision.transforms.functional.InterpolationMode.NEAREST: 'nearest', fill: Optional[List[float]] = None*)

Bases: `torchvision.transforms.autoaugment.AutoAugment`

**forward**(*results*)

*img* (PIL Image or Tensor): Image to be transformed.

**Returns** AutoAugmented image.

**Return type** PIL Image or Tensor

**training:** `bool`

**class** `easycv.datasets.shared.pipelines.third_transforms_wrapper.CenterCrop`(*size*)

Bases: `torchvision.transforms.transforms.CenterCrop`

**forward**(*results*)

**Parameters** *img* (PIL Image or Tensor) – Image to be cropped.

**Returns** Cropped image.

**Return type** PIL Image or Tensor

**training:** `bool`

**class** `easycv.datasets.shared.pipelines.third_transforms_wrapper.ColorJitter`(*brightness=0, contrast=0, saturation=0, hue=0*)

Bases: `torchvision.transforms.transforms.ColorJitter`

**forward**(*results*)

**Parameters** *img* (PIL Image or Tensor) – Input image.

**Returns** Color jittered image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**ConvertImageDtype**(dtype: *torch.dtype*)

Bases: torchvision.transforms.transforms.ConvertImageDtype

**forward**(results)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**FiveCrop**(size)

Bases: torchvision.transforms.transforms.FiveCrop

**forward**(results)

**Parameters** **img** (*PIL Image or Tensor*) – Image to be cropped.

**Returns** tuple of 5 images. Image can be PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**GaussianBlur**(kernel\_size, sigma=(0.1, 2.0))

Bases: torchvision.transforms.transforms.GaussianBlur

**forward**(results)

**Parameters** **img** (*PIL Image or Tensor*) – image to be blurred.

**Returns** Gaussian blurred image

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**Grayscale**(num\_output\_channels=1)

Bases: torchvision.transforms.transforms.Grayscale

**forward**(results)

**Parameters** **img** (*PIL Image or Tensor*) – Image to be converted to grayscale.

**Returns** Grayscaled image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**Lambda**(lambda)

Bases: torchvision.transforms.transforms.Lambda

```
class easycv.datasets.shared.pipelines.third_transforms_wrapper.LinearTransformation(transformation_matrix,
 mean_vector)
 Bases: torchvision.transforms.transforms.LinearTransformation
 forward(results)

 Parameters tensor (Tensor) – Tensor image to be whitened.
 Returns Transformed image.
 Return type Tensor
 training: bool
class easycv.datasets.shared.pipelines.third_transforms_wrapper.Normalize(mean, std,
 inplace=False)
 Bases: torchvision.transforms.transforms.Normalize
 forward(results)

 Parameters tensor (Tensor) – Tensor image to be normalized.
 Returns Normalized Tensor image.
 Return type Tensor
 training: bool
class easycv.datasets.shared.pipelines.third_transforms_wrapper.PILToTensor
 Bases: torchvision.transforms.transforms.PILToTensor
class easycv.datasets.shared.pipelines.third_transforms_wrapper.Pad(padding, fill=0,
 padding_mode='constant')
 Bases: torchvision.transforms.transforms.Pad
 forward(results)

 Parameters img (PIL Image or Tensor) – Image to be padded.
 Returns Padded image.
 Return type PIL Image or Tensor
 training: bool
```

```

class easycv.datasets.shared.pipelines.third_transforms_wrapper.RandAugment(num_ops: int =
 2, magnitude: int
 = 9,
 num_magnitude_bins:
 int = 31,
 interpolation:
 torchvision.transforms.functional.InterpolationMode.NEAREST,
 fill:
 Optional[List[float]]
 = None)

```

Bases: torchvision.transforms.autoaugment.RandAugment

**forward**(results)

img (PIL Image or Tensor): Image to be transformed.

**Returns** Transformed image.

**Return type** PIL Image or Tensor

**training:** bool

```

class easycv.datasets.shared.pipelines.third_transforms_wrapper.RandomAdjustSharpness(sharpness_factor,
 p=0.5)

```

Bases: torchvision.transforms.transforms.RandomAdjustSharpness

**forward**(results)

**Parameters** **img** (PIL Image or Tensor) – Image to be sharpened.

**Returns** Randomly sharpened image.

**Return type** PIL Image or Tensor

**training:** bool

```

class easycv.datasets.shared.pipelines.third_transforms_wrapper.RandomAffine(degrees,
 translate=None,
 scale=None,
 shear=None,
 interpolation=<InterpolationMode.NEAREST>,
 fill=0,
 fillcolor=None,
 resample=None,
 center=None)

```

Bases: torchvision.transforms.transforms.RandomAffine

**forward**(results)

img (PIL Image or Tensor): Image to be transformed.

**Returns** Affine transformed image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomAutocontrast**(*p=0.5*)  
Bases: torchvision.transforms.transforms.RandomAutocontrast

**forward**(*results*)

**Parameters** *img* (*PIL Image or Tensor*) – Image to be autocontrasted.

**Returns** Randomly autocontrasted image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomChoice**(*transforms*,  
*p=None*)

Bases: torchvision.transforms.transforms.RandomChoice

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomCrop**(*size*,  
*padding=None*,  
*pad\_if\_needed=False*,  
*fill=0*,  
*padding\_mode='constant'*)

Bases: torchvision.transforms.transforms.RandomCrop

**forward**(*results*)

**Parameters** *img* (*PIL Image or Tensor*) – Image to be cropped.

**Returns** Cropped image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomEqualize**(*p=0.5*)  
Bases: torchvision.transforms.transforms.RandomEqualize

**forward**(*results*)

**Parameters** *img* (*PIL Image or Tensor*) – Image to be equalized.

**Returns** Randomly equalized image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomErasing**(*p=0.5*,  
*scale=(0.02*,  
*0.33)*,  
*ratio=(0.3*,  
*3.3)*, *value=0*,  
*in-*  
*place=False*)

Bases: torchvision.transforms.transforms.RandomErasing

**forward**(*results*)

**Parameters** *img* (*Tensor*) – Tensor image to be erased.



**Returns** Erased Tensor image.

**Return type** img (Tensor)

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomGrayscale**(*p=0.1*)

Bases: torchvision.transforms.transforms.RandomGrayscale

**forward**(*results*)

**Parameters** **img** (*PIL Image or Tensor*) – Image to be converted to grayscale.

**Returns** Randomly grayscaled image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomHorizontalFlip**(*p=0.5*)

Bases: torchvision.transforms.transforms.RandomHorizontalFlip

**forward**(*results*)

**Parameters** **img** (*PIL Image or Tensor*) – Image to be flipped.

**Returns** Randomly flipped image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomInvert**(*p=0.5*)

Bases: torchvision.transforms.transforms.RandomInvert

**forward**(*results*)

**Parameters** **img** (*PIL Image or Tensor*) – Image to be inverted.

**Returns** Randomly color inverted image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomOrder**(*transforms*)

Bases: torchvision.transforms.transforms.RandomOrder

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomPerspective**(*distortion\_scale=0.5,*

*p=0.5,*

*interpo-*

*la-*

*tion=<InterpolationMode.B*

*'bilin-*

*ear'>,*

*fill=0)*

Bases: torchvision.transforms.transforms.RandomPerspective

**forward**(*results*)

**Parameters** **img** (*PIL Image or Tensor*) – Image to be Perspectively transformed.

**Returns** Randomly transformed image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomPosterize**(bits, p=0.5)

Bases: torchvision.transforms.transforms.RandomPosterize

**forward**(results)

**Parameters** **img** (PIL Image or Tensor) – Image to be posterized.

**Returns** Randomly posterized image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomResizedCrop**(size, scale=(0.08, 1.0), ratio=(0.75, 1.3333333333333333), interpolation=<InterpolationMode.BILINEAR>)

Bases: torchvision.transforms.transforms.RandomResizedCrop

**forward**(results)

**Parameters** **img** (PIL Image or Tensor) – Image to be cropped and resized.

**Returns** Randomly cropped and resized image.

**Return type** PIL Image or Tensor

**training:** bool

**class** easycv.datasets.shared.pipelines.third\_transforms\_wrapper.**RandomRotation**(degrees, interpolation=<InterpolationMode.NEAREST>, expand=False, center=None, fill=0, resample=None)

Bases: torchvision.transforms.transforms.RandomRotation

**forward**(results)

**Parameters** **img** (PIL Image or Tensor) – Image to be rotated.

**Returns** Rotated image.

**Return type** PIL Image or Tensor

**training:** bool

---

```
class easycv.datasets.shared.pipelines.third_transforms_wrapper.RandomSolarize(threshold,
 p=0.5)
```

```
 Bases: torchvision.transforms.transforms.RandomSolarize
```

```
 forward(results)
```

```
 Parameters img (PIL Image or Tensor) – Image to be solarized.
```

```
 Returns Randomly solarized image.
```

```
 Return type PIL Image or Tensor
```

```
 training: bool
```

```
class easycv.datasets.shared.pipelines.third_transforms_wrapper.RandomVerticalFlip(p=0.5)
```

```
 Bases: torchvision.transforms.transforms.RandomVerticalFlip
```

```
 forward(results)
```

```
 Parameters img (PIL Image or Tensor) – Image to be flipped.
```

```
 Returns Randomly flipped image.
```

```
 Return type PIL Image or Tensor
```

```
 training: bool
```

```
class easycv.datasets.shared.pipelines.third_transforms_wrapper.Resize(size, interpolation=<InterpolationMode.BILINEAR: 'bilinear'>,
 max_size=None,
 antialias=None)
```

```
 Bases: torchvision.transforms.transforms.Resize
```

```
 forward(results)
```

```
 Parameters img (PIL Image or Tensor) – Image to be scaled.
```

```
 Returns Rescaled image.
```

```
 Return type PIL Image or Tensor
```

```
 training: bool
```

```
class easycv.datasets.shared.pipelines.third_transforms_wrapper.TenCrop(size,
 vertical_flip=False)
```

```
 Bases: torchvision.transforms.transforms.TenCrop
```

```
 forward(results)
```

```
 Parameters img (PIL Image or Tensor) – Image to be cropped.
```

```
 Returns tuple of 10 images. Image can be PIL Image or Tensor
```

```
 training: bool
```

```
class easycv.datasets.shared.pipelines.third_transforms_wrapper.ToPILImage(mode=None)
```

```
 Bases: torchvision.transforms.transforms.ToPILImage
```

```
class easycv.datasets.shared.pipelines.third_transforms_wrapper.ToTensor
```

```
 Bases: torchvision.transforms.transforms.ToTensor
```

```
class easycv.datasets.shared.pipelines.third_transforms_wrapper.TrivialAugmentWide(num_magnitude_bins:
 int =
 31,
 interpo-
 lation:
 torchvision.transforms.functional.
 = <In-
 terpol-
 ation-
 Mode.NEAREST:
 'near-
 est'>,
 fill: Op-
 tional[List[float]]
 =
 None)

Bases: torchvision.transforms.autoaugment.TrivialAugmentWide

forward(results)
 img (PIL Image or Tensor): Image to be transformed.

 Returns Transformed image.

 Return type PIL Image or Tensor

training: bool
```

### easycv.datasets.shared.pipelines.transforms module

```
class easycv.datasets.shared.pipelines.transforms.Compose(transforms, profiling=False)
 Bases: object

 Compose a data pipeline with a sequence of transforms. :param transforms: Either config dicts of transforms or
 transform objects. :type transforms: list[dict | callable]

 __init__(transforms, profiling=False)
 Initialize self. See help(type(self)) for accurate signature.
```

### Submodules

#### easycv.datasets.shared.base module

```
class easycv.datasets.shared.base.BaseDataset(data_source, pipeline, profiling=False)
 Bases: Generic[torch.utils.data.dataset.T_co]

 Base Dataset

 __init__(data_source, pipeline, profiling=False)
 Initialize self. See help(type(self)) for accurate signature.

 abstract evaluate(results, evaluators, logger=None, **kwargs)

 visualize(results, **kwargs)
 Visualize the model output results on validation data. Returns: A dictionary

 If add image visualization, return dict containing images: List of visualized images.
 img_metas: List of length number of test images,
```

dict of image meta info, containing filename, img\_shape, origin\_img\_shape, scale\_factor and so on.

### easyencv.datasets.shared.dali\_tfrecord\_imagenet module

**class** easyencv.datasets.shared.dali\_tfrecord\_imagenet.**DaliLoaderWrapper**(*dali\_loader, batch\_size, label\_offset=0*)

Bases: object

**\_\_init\_\_**(*dali\_loader, batch\_size, label\_offset=0*)  
Initialize self. See help(type(self)) for accurate signature.

**evaluate**(*results, evaluators, logger=None*)  
evaluate classification task

#### Parameters

- **results** – a dict of list of tensor, including prediction and groundtruth info, where prediction tensor is NxCan and the same with groundtruth labels.
- **evaluators** – a list of evaluator

**Returns** a dict of float, different metric values

**Return type** eval\_result

**class** easyencv.datasets.shared.dali\_tfrecord\_imagenet.**ImageNetTFRecordPipe**(*data\_source, transforms, batch\_size, distributed, random\_shuffle=True, workers\_per\_gpu=2, device='gpu'*)

Bases: nvidia.dali.pipeline.Pipeline

**\_\_init\_\_**(*data\_source, transforms, batch\_size, distributed, random\_shuffle=True, workers\_per\_gpu=2, device='gpu'*)  
Initialize self. See help(type(self)) for accurate signature.

**define\_graph**()  
This function is defined by the user to construct the graph of operations for their pipeline.  
It returns a list of outputs created by calling DALI Operators.

**class** easyencv.datasets.shared.dali\_tfrecord\_imagenet.**DaliImageNetTFRecordDataSet**(*data\_source, pipeline, distributed, batch\_size, label\_offset=0, random\_shuffle=True, workers\_per\_gpu=2*)

Bases: object

**\_\_init\_\_**(*data\_source, pipeline, distributed, batch\_size, label\_offset=0, random\_shuffle=True, workers\_per\_gpu=2*)  
Initialize self. See help(type(self)) for accurate signature.

`get_dataloader()`

### `easycv.datasets.shared.dali_tfrecord_multi_view` module

```
class easycv.datasets.shared.dali_tfrecord_multi_view.DaliLoaderWrapper(dali_loader,
 batch_size,
 return_list)
```

Bases: object

```
__init__(dali_loader, batch_size, return_list)
 Initialize self. See help(type(self)) for accurate signature.
```

```
class easycv.datasets.shared.dali_tfrecord_multi_view.DaliTFRecordMultiViewPipe(data_source,
 trans-
 forms_list,
 batch_size,
 distributed,
 ran-
 dom_shuffle=True,
 work-
 ers_per_gpu=2,
 de-
 vice='gpu')
```

Bases: `nvidia.dali.pipeline.Pipeline`

```
__init__(data_source, transforms_list, batch_size, distributed, random_shuffle=True, workers_per_gpu=2,
 device='gpu')
 Initialize self. See help(type(self)) for accurate signature.
```

```
define_graph()
 This function is defined by the user to construct the graph of operations for their pipeline.
 It returns a list of outputs created by calling DALI Operators.
```

```
class easycv.datasets.shared.dali_tfrecord_multi_view.DaliTFRecordMultiViewDataset(data_source,
 num_views,
 pipelines,
 dis-
 tributed,
 batch_size,
 ran-
 dom_shuffle=True,
 work-
 ers_per_gpu=2)
```

Bases: object

Adapt to dali, the dataset outputs multiple views of an image. The number of views in the output dict depends on *num\_views*. The image can be processed by one pipeline or multiple pipelines. :param *num\_views*: The number of different views. :type *num\_views*: list :param *pipelines*: A list of pipelines. :type *pipelines*: list[list[dict]]

```
__init__(data_source, num_views, pipelines, distributed, batch_size, random_shuffle=True,
 workers_per_gpu=2)
 Initialize self. See help(type(self)) for accurate signature.
```

`get_dataloader()`

**easycv.datasets.shared.dataset\_wrappers module**

**class** easycv.datasets.shared.dataset\_wrappers.**ConcatDataset**(*datasets*)

Bases: torch.utils.data.dataset.Dataset[torch.utils.data.dataset.T\_co]

A wrapper of concatenated dataset.

Same as torch.utils.data.dataset.ConcatDataset, but concat the group flag for image aspect ratio.

**Parameters** **datasets** (list[Dataset]) – A list of datasets.

**\_\_init\_\_**(*datasets*)

Initialize self. See help(type(self)) for accurate signature.

**datasets:** List[torch.utils.data.dataset.Dataset[torch.utils.data.dataset.T\_co]]

**cumulative\_sizes:** List[int]

**class** easycv.datasets.shared.dataset\_wrappers.**RepeatDataset**(*dataset, times*)

Bases: object

A wrapper of repeated dataset.

The length of repeated dataset will be *times* larger than the original dataset. This is useful when the data loading time is long but the dataset is small. Using RepeatDataset can reduce the data loading time between epochs.

**Parameters**

- **dataset** (Dataset) – The dataset to be repeated.
- **times** (int) – Repeat times.

**\_\_init\_\_**(*dataset, times*)

Initialize self. See help(type(self)) for accurate signature.

**easycv.datasets.shared.multi\_view module**

**class** easycv.datasets.shared.multi\_view.**MultiViewDataset**(*data\_source, num\_views, pipelines*)

Bases: Generic[torch.utils.data.dataset.T\_co]

The dataset outputs multiple views of an image. The number of views in the output dict depends on *num\_views*. The image can be processed by one pipeline or multiple pipelines. :param num\_views: The number of different views. :type num\_views: list :param pipelines: A list of pipelines. :type pipelines: list[list[dict]]

**\_\_init\_\_**(*data\_source, num\_views, pipelines*)

Initialize self. See help(type(self)) for accurate signature.

**evaluate**(*results, evaluators, logger=None*)

**easycv.datasets.shared.odps\_reader module**

easycv.datasets.shared.odps\_reader.**set\_dataloader\_workid**(*value*)

easycv.datasets.shared.odps\_reader.**set\_dataloader\_worknum**(*value*)

easycv.datasets.shared.odps\_reader.**get\_dist\_image**(*img\_url, max\_try=10*)

**class** easycv.datasets.shared.odps\_reader.**OdpsReader**(*table\_name, selected\_cols=[],  
excluded\_cols=[], random\_start=False,  
odps\_io\_config=None, image\_col=['url\_image'],  
image\_type=['url']*)

Bases: object

```
__init__(table_name, selected_cols=[], excluded_cols=[], random_start=False, odps_io_config=None,
 image_col=['url_image'], image_type=['url'])
```

Init odps reader and datasource set to load data from odps table

#### Parameters

- **table\_name** (*str*) – odps table to load
- **selected\_cols** (*list(str)*) – select column
- **excluded\_cols** (*list(str)*) – exclude column
- **random\_start** (*bool*) – random start for odps table
- **odps\_io\_config** (*dict*) – odps config contains access\_id, access\_key, endpoint
- **image\_col** (*list(str)*) – image column names
- **image\_type** (*list(str)*) – image column types support url/base64, must be same length with image type or 0

**Returns :** None

```
get_length()
```

```
reset_reader(dataloader_workid, dataloader_worknum)
```

```
get_sample(idx)
```

```
b64_decode()
```

### easycv.datasets.shared.raw module

```
class easycv.datasets.shared.raw.RawDataset(data_source, pipeline)
```

Bases: Generic[torch.utils.data.dataset.T\_co]

```
__init__(data_source, pipeline)
```

Initialize self. See help(type(self)) for accurate signature.

```
evaluate(scores, keyword, logger=None)
```

## 12.1.7 easycv.datasets.utils package

### Submodules

#### easycv.datasets.utils.tfrecord\_util module

```
easycv.datasets.utils.tfrecord_util.get_path_and_index(file_list_or_path)
```

```
easycv.datasets.utils.tfrecord_util.download_tfrecord(file_list_or_path, target_path, slice_count=1,
 slice_id=0, force=False)
```

Download data from oss. Use the processes on the gpus to slice download, each gpu process downloads part of the data. The number of slices is the same as the number of gpu processes. Support tfrecord of ImageNet style. tfrecord\_dir

```
|—train1 |—train1.idx |—train2 |—train2.idx |—...
```

#### Parameters



- **file\_list\_or\_path** – A list of absolute data path or a path str type(file\_list) == list means this is the list type(file\_list) == str means open(file\_list).readlines()
- **target\_path** – A str, download path
- **slice\_count** – Download worker num
- **slice\_id** – Download worker ID
- **force** – If false, skip download if the file already exists in the target path. If true, recopy and replace the original file.

**Returns** list of str, download tfrecord path index\_path: list of str, download tfrecord idx path

**Return type** path

#### **easycv.datasets.utils.type\_util module**

`easycv.datasets.utils.type_util.is_dali_dataset_type(type_name)`

## **12.2 Submodules**

### **12.3 easycv.datasets.builder module**

`easycv.datasets.builder.build_dataset(cfg, default_args=None)`

`easycv.datasets.builder.build_dali_dataset(cfg, default_args=None)`

`easycv.datasets.builder.build_datasource(cfg)`

### **12.4 easycv.datasets.registry module**



## EASYCV.HOOKS PACKAGE

### 13.1 Submodules

### 13.2 `easycv.hooks.best_ckpt_saver_hook` module

**class** `easycv.hooks.best_ckpt_saver_hook.BestCkptSaverHook`(*by\_epoch=True, save\_optimizer=True, best\_metric\_name=[], best\_metric\_type=[], \*\*kwargs*)

Bases: `mmdcv.runner.hooks.hook.Hook`

Save checkpoints periodically.

#### Parameters

- **by\_epoch** (*bool*) – Saving checkpoints by epoch or by iteration. Default: True.
- **save\_optimizer** (*bool*) – Whether to save optimizer state\_dict in the checkpoint. It is usually used for resuming experiments. Default: True.
- **best\_metric\_name** (*List(str)*) – metric name to save best, such as “neck\_top1”... Default: [], do not save anything
- **best\_metric\_type** (*List(str)*) – metric type to define best, should be “max”, “min” if `len(best_metric_type) <= len(best_metric_name)`, use “max” to append.

**\_\_init\_\_**(*by\_epoch=True, save\_optimizer=True, best\_metric\_name=[], best\_metric\_type=[], \*\*kwargs*)  
Initialize self. See `help(type(self))` for accurate signature.

**before\_run**(*runner*)

**after\_train\_epoch**(*runner*)

### 13.3 `easycv.hooks.builder` module

`easycv.hooks.builder.build_hook`(*cfg, default\_args=None*)

## 13.4 easycv.hooks.byol\_hook module

**class** easycv.hooks.byol\_hook.**BYOLHook**(*end\_momentum=1.0, \*\*kwargs*)

Bases: `mmcv.runner.hooks.hook.Hook`

Hook in BYOL

**This hook including momentum adjustment in BYOL following:**  $m = 1 - (1 - m_0) * (\cos(\pi * k / K) + 1) / 2$ ,  $k$ : current step,  $K$ : total steps.

**\_\_init\_\_**(*end\_momentum=1.0, \*\*kwargs*)

Initialize self. See `help(type(self))` for accurate signature.

**before\_train\_iter**(*runner*)

## 13.5 easycv.hooks.dino\_hook module

`easycv.hooks.dino_hook.cosine_scheduler`(*base\_value, final\_value, epochs, niter\_per\_ep, warmup\_epochs=0, start\_warmup\_value=0*)

**class** easycv.hooks.dino\_hook.**DINOHook**(*momentum\_teacher=0.996, weight\_decay=0.04, weight\_decay\_end=0.4, \*\*kwargs*)

Bases: `mmcv.runner.hooks.hook.Hook`

Hook in DINO

**\_\_init\_\_**(*momentum\_teacher=0.996, weight\_decay=0.04, weight\_decay\_end=0.4, \*\*kwargs*)

Initialize self. See `help(type(self))` for accurate signature.

**before\_run**(*runner*)

**before\_train\_iter**(*runner*)

**after\_train\_iter**(*runner*)

**before\_train\_epoch**(*runner*)

## 13.6 easycv.hooks.ema\_hook module

**class** easycv.hooks.ema\_hook.**ModelEMA**(*model, decay=0.9999, updates=0*)

Bases: `object`

Model Exponential Moving Average from <https://github.com/rwightman/pytorch-image-models> Keep a moving average of everything in the model `state_dict` (parameters and buffers). This is intended to allow functionality like [https://www.tensorflow.org/api\\_docs/python/tf/train/ExponentialMovingAverage](https://www.tensorflow.org/api_docs/python/tf/train/ExponentialMovingAverage) A smoothed version of the weights is necessary for some training schemes to perform well. This class is sensitive where it is initialized in the sequence of model init, GPU assignment and distributed training wrappers.

In Yolo5s, ema help increase mAP from 0.27 to 0.353

**\_\_init\_\_**(*model, decay=0.9999, updates=0*)

Initialize self. See `help(type(self))` for accurate signature.

**update**(*model*)

**update\_attr**(*model, include=(), exclude=('process\_group', 'reducer')*)

```
class easycv.hooks.ema_hook.EMAHook(decay=0.9999, copy_model_attr=())
```

Bases: `mmcv.runner.hooks.hook.Hook`

Hook to carry out Exponential Moving Average

```
__init__(decay=0.9999, copy_model_attr=())
```

#### Parameters

- **decay** – decay rate for exponential moving average
- **copy\_model\_attr** – attribute to copy from origin model to ema model

```
before_run(runner)
```

```
before_train_epoch(runner)
```

```
after_train_iter(runner)
```

## 13.7 easycv.hooks.eval\_hook module

```
class easycv.hooks.eval_hook.EvalHook(dataloader, initial=False, interval=1, mode='test',
 flush_buffer=True, **eval_kwargs)
```

Bases: `mmcv.runner.hooks.hook.Hook`

Evaluation hook.

**dataloader**

A PyTorch dataloader.

**Type** `DataLoader`

**interval**

Evaluation interval (by epochs). Default: 1.

**Type** `int`

**mode**

model forward mode

**Type** `str`

**flush\_buffer**

flush log buffer

**Type** `bool`

```
__init__(dataloader, initial=False, interval=1, mode='test', flush_buffer=True, **eval_kwargs)
```

Initialize self. See `help(type(self))` for accurate signature.

```
before_run(runner)
```

```
after_train_epoch(runner)
```

```
add_visualization_info(runner, results)
```

```
evaluate(runner, results)
```

```
class easycv.hooks.eval_hook.DistEvalHook(dataloader, interval=1, mode='test', initial=False,
 gpu_collect=False, flush_buffer=True, **eval_kwargs)
```

Bases: `easycv.hooks.eval_hook.EvalHook`

Distributed evaluation hook.

**dataloader**

A PyTorch dataloader.

**Type** DataLoader

**interval**

Evaluation interval (by epochs). Default: 1.

**Type** int

**mode**

model forward mode

**Type** str

**tmpdir**

Temporary directory to save the results of all processes. Default: None.

**Type** str | None

**gpu\_collect**

Whether to use gpu or cpu to collect results. Default: False.

**Type** bool

**\_\_init\_\_**(*dataloader*, *interval*=1, *mode*='test', *initial*=False, *gpu\_collect*=False, *flush\_buffer*=True, *\*\*eval\_kwargs*)

Initialize self. See help(type(self)) for accurate signature.

**before\_run**(*runner*)

**after\_train\_epoch**(*runner*)

## 13.8 easycv.hooks.export\_hook module

**class** easycv.hooks.export\_hook.**ExportHook**(*cfg*, *ckpt\_filename\_tmpl*='epoch\_{}.pth',  
*export\_ckpt\_filename\_tmpl*='epoch\_{}\_export.pt',  
*export\_after\_each\_ckpt*=False)

Bases: `mmcv.runner.hooks.hook.Hook`

export model when training on pai

**\_\_init\_\_**(*cfg*, *ckpt\_filename\_tmpl*='epoch\_{}.pth', *export\_ckpt\_filename\_tmpl*='epoch\_{}\_export.pt',  
*export\_after\_each\_ckpt*=False)

**Parameters**

- **cfg** – config dict
- **ckpt\_filename\_tmpl** – checkpoint filename template

**export\_model**(*runner*, *epoch*)

**after\_train\_iter**(*runner*)

**after\_train\_epoch**(*runner*)

**after\_run**(*runner*)

## 13.9 easycv.hooks.extractor module

**class** easycv.hooks.extractor.**Extractor**(dataset, imgs\_per\_gpu, workers\_per\_gpu, dist\_mode=False)  
 Bases: object

**\_\_init\_\_**(dataset, imgs\_per\_gpu, workers\_per\_gpu, dist\_mode=False)  
 Initialize self. See help(type(self)) for accurate signature.

## 13.10 easycv.hooks.optimizer\_hook module

**class** easycv.hooks.optimizer\_hook.**OptimizerHook**(update\_interval=1, grad\_clip=None, coalesce=True, bucket\_size\_mb=-1, ignore\_key=[], ignore\_key\_epoch=[], multiply\_key=[], multiply\_rate=[])  
 Bases: mmdcv.runner.hooks.optimizer.OptimizerHook

**\_\_init\_\_**(update\_interval=1, grad\_clip=None, coalesce=True, bucket\_size\_mb=-1, ignore\_key=[], ignore\_key\_epoch=[], multiply\_key=[], multiply\_rate=[])  
 ignore\_key: [str,...], ignore\_key[i], name of parameters, which's gradient will be set to zero before every optimizer step when epoch < ignore\_key\_epoch[i] ignore\_key\_epoch: [int,...], epoch < ignore\_key\_epoch[i], ignore\_key[i]'s gradient will be set to zero.  
 multiply\_key:[str,...] multiply\_key[i], name of parameters, which will set different learning rate ratio by multiply\_rate multiply\_rate:[float,...] multiply\_rate[i], different ratio

**before\_run**(runner)

**after\_train\_iter**(runner)

**class** easycv.hooks.optimizer\_hook.**AMPFP16OptimizerHook**(update\_interval=1, grad\_clip=None, coalesce=True, bucket\_size\_mb=-1, ignore\_key=[], ignore\_key\_epoch=[])  
 Bases: [easycv.hooks.optimizer\\_hook.OptimizerHook](#)

**\_\_init\_\_**(update\_interval=1, grad\_clip=None, coalesce=True, bucket\_size\_mb=-1, ignore\_key=[], ignore\_key\_epoch=[])  
 ignore\_key: [str,...], ignore\_key[i], name of parameters, which's gradient will be set to zero before every optimizer step when epoch < ignore\_key\_epoch[i] ignore\_key\_epoch: [int,...], epoch < ignore\_key\_epoch[i], ignore\_key[i]'s gradient will be set to zero.

**before\_run**(runner)

**after\_train\_iter**(runner)

## 13.11 easycv.hooks.oss\_sync\_hook module

**class** easycv.hooks.oss\_sync\_hook.**OSSSyncHook**(work\_dir, oss\_work\_dir, interval=1, ckpt\_filename\_tmpl='epoch\_{}.pth', export\_ckpt\_filename\_tmpl='epoch\_{}\_export.pt', other\_file\_list=[], iter\_interval=None)  
 Bases: mmdcv.runner.hooks.hook.Hook

upload log files and checkpoints to oss when training on pai

```
__init__(work_dir, oss_work_dir, interval=1, ckpt_filename_tmpl='epoch_{}.pth',
 export_ckpt_filename_tmpl='epoch_{}_export.pt', other_file_list=[], iter_interval=None)
```

#### Parameters

- **work\_dir** – work\_dir in cfg
- **oss\_work\_dir** – oss directory where to upload local files in work\_dir
- **interval** – upload frequency
- **ckpt\_filename\_tmpl** – checkpoint filename template
- **other\_file\_list** – other file need to be upload to oss
- **iter\_interval** – upload frequency by iter interval, default to be None, means do it with certain assignment

```
upload_file(runner)
```

```
after_train_iter(runner)
```

```
after_train_epoch(runner)
```

```
after_run(runner)
```

## 13.12 easycv.hooks.registry module

## 13.13 easycv.hooks.show\_time\_hook module

```
class easycv.hooks.show_time_hook.TIMEHook(end_momentum=1.0, **kwargs)
```

Bases: `mmcv.runner.hooks.hook.Hook`

This hook to show time for runner running process

```
__init__(end_momentum=1.0, **kwargs)
```

Initialize self. See `help(type(self))` for accurate signature.

```
before_train_iter(runner)
```

```
after_train_iter(runner)
```

## 13.14 easycv.hooks.swav\_hook module

```
class easycv.hooks.swav_hook.SWAVHook(gpu_batch_size=32, dump_path='data', **kwargs)
```

Bases: `mmcv.runner.hooks.hook.Hook`

Hook in SWAV

```
__init__(gpu_batch_size=32, dump_path='data', **kwargs)
```

Initialize self. See `help(type(self))` for accurate signature.

```
before_run(runner)
```

```
before_train_epoch(runner)
```

```
after_train_epoch(runner)
```



## 13.15 easycv.hooks.sync\_norm\_hook module

`easycv.hooks.sync_norm_hook.get_norm_states(module)`

**class** `easycv.hooks.sync_norm_hook.SyncNormHook`(*no\_aug\_epochs=15, interval=1, \*\*kwargs*)

Bases: `mmcv.runner.hooks.hook.Hook`

Synchronize Norm states after training epoch, currently used in YOLOX.

### Parameters

- **no\_aug\_epochs** (*int*) – The number of latter epochs in the end of the training to switch to synchronizing norm interval. Default: 15.
- **interval** (*int*) – Synchronizing norm interval. Default: 1.

**\_\_init\_\_**(*no\_aug\_epochs=15, interval=1, \*\*kwargs*)

Initialize self. See `help(type(self))` for accurate signature.

**before\_train\_epoch**(*runner*)

**after\_train\_epoch**(*runner*)

Synchronizing norm.

## 13.16 easycv.hooks.sync\_random\_size\_hook module

**class** `easycv.hooks.sync_random_size_hook.SyncRandomSizeHook`(*ratio\_range=(14, 26),  
img\_scale=(640, 640), interval=10,  
device='cuda', \*\*kwargs*)

Bases: `mmcv.runner.hooks.hook.Hook`

Change and synchronize the random image size across ranks, currently used in YOLOX.

### Parameters

- **ratio\_range** (*tuple[int]*) – Random ratio range. It will be multiplied by 32, and then change the dataset output image size. Default: (14, 26).
- **img\_scale** (*tuple[int]*) – Size of input image. Default: (640, 640).
- **interval** (*int*) – The interval of change image size. Default: 10.
- **device** (*torch.device | str*) – device for returned tensors. Default: 'cuda'.

**\_\_init\_\_**(*ratio\_range=(14, 26), img\_scale=(640, 640), interval=10, device='cuda', \*\*kwargs*)

Initialize self. See `help(type(self))` for accurate signature.

**after\_train\_iter**(*runner*)

Change the dataset output image size.

## 13.17 easycv.hooks.tensorboard module

```
class easycv.hooks.tensorboard.TensorboardLoggerHookV2(log_dir=None, interval=10,
 ignore_last=True, reset_flag=False,
 by_epoch=True)
 Bases: mmcv.runner.hooks.logger.tensorboard.TensorboardLoggerHook
 visualization_log(runner)
 Images Visualization. visualization_buffer is a dictionary containing:
 images (list): list of visualized images. img metas (list of dict, optional): dict containing
 ori_filename and so on.
 ori_filename will be displayed as the tag of the image by default.
 log(runner)
 after_train_iter(runner)
```

## 13.18 easycv.hooks.wandb module

```
class easycv.hooks.wandb.WandbLoggerHookV2(init_kwargs=None, interval=10, ignore_last=True,
 reset_flag=False, commit=True, by_epoch=True,
 with_step=True)
 Bases: mmcv.runner.hooks.logger.wandb.WandbLoggerHook
 visualization_log(runner)
 Images Visualization. visualization_buffer is a dictionary containing:
 images (list): list of visualized images. img metas (list of dict, optional): dict containing
 ori_filename and so on.
 ori_filename will be displayed as the tag of the image by default.
 log(runner)
 after_train_iter(runner)
```

## 13.19 easycv.hooks.yolox\_lr\_hook module

```
class easycv.hooks.yolox_lr_hook.YOLOXLRUpdaterHook(num_last_epochs, **kwargs)
 Bases: mmcv.runner.hooks.lr_updater.CosineAnnealingLRUpdaterHook
```

YOLOX learning rate scheme.

There are two main differences between YOLOXLRUpdaterHook and CosineAnnealingLRUpdaterHook.

1. **When the current running epoch is greater than** *max\_epoch-last\_epoch*, a fixed learning rate will be used
2. The exp warmup scheme is different with LrUpdaterHook in MMCV

**Parameters** *num\_last\_epochs* (*int*) – The number of epochs with a fixed learning rate before the end of the training.

```
__init__(num_last_epochs, **kwargs)
 Initialize self. See help(type(self)) for accurate signature.
```

```
get_warmup_lr(cur_iters)
```

```
get_lr(runner, base_lr)
```

## 13.20 easycv.hooks.yolox\_mode\_switch\_hook module

```
class easycv.hooks.yolox_mode_switch_hook.YOLOXModeSwitchHook(no_aug_epochs=15,
 skip_type_keys=('MMMosaic',
 'MMRandomAffine', 'MMMixUp'),
 **kwargs)
```

Bases: `mmcv.runner.hooks.hook.Hook`

Switch the mode of YOLOX during training.

This hook turns off the mosaic and mixup data augmentation and switches to use L1 loss in bbox\_head.

**Parameters** **no\_aug\_epochs** – The number of latter epochs in the end of the training to close the data augmentation and switch to L1 loss. Default: 15.

**\_\_init\_\_**(*no\_aug\_epochs=15*, *skip\_type\_keys*=('MMMosaic', 'MMRandomAffine', 'MMMixUp'), *\*\*kwargs*)  
Initialize self. See help(type(self)) for accurate signature.

**before\_train\_epoch**(*runner*)

Close mosaic and mixup augmentation and switches to use L1 loss.



## EASYCV.PREDICTORS PACKAGE

### 14.1 Submodules

### 14.2 `easycv.predictors.base` module

**class** `easycv.predictors.base.NumpyToPIL`

Bases: `object`

**class** `easycv.predictors.base.Predictor(model_path, numpy_to_pil=True)`

Bases: `object`

**\_\_init\_\_**(*model\_path*, *numpy\_to\_pil=True*)

Initialize self. See `help(type(self))` for accurate signature.

**preprocess**(*image\_list*)

**predict\_batch**(*image\_batch*, *\*\*forward\_kwargs*)

predict using batched data

**Parameters**

- **image\_batch** (*torch.Tensor*) – tensor with shape [N, 3, H, W]
- **forward\_kwargs** – kwargs for additional parameters

**Returns** the output of `model.forward`, list or tuple

**Return type** `output`

### 14.3 `easycv.predictors.builder` module

`easycv.predictors.builder.build_predictor(cfg)`

## 14.4 easycv.predictors.classifier module

**class** easycv.predictors.classifier.TorchClassifier(*model\_path*, *model\_config*=None, *topk*=1,  
label\_map\_path=None)

Bases: *easycv.predictors.interface.PredictorInterface*

**\_\_init\_\_**(*model\_path*, *model\_config*=None, *topk*=1, *label\_map\_path*=None)  
init model

### Parameters

- **model\_path** – model file path
- **model\_config** – config string for model to init, in json format

### get\_output\_type()

in this function user should return a type dict, which indicates which type of data should the output of predictor be converted to \* type json, data will be serialized to json str

- type image, data will be converted to encode image binary and write to oss file, whose name is output\_dir/\${key}/\${input\_filename}\_\${idx}.jpg, where input\_filename is the base filename extracted from url, key corresponds to the key in the dict of output\_type, if the type of data indexed by key is a list, idx is the index of element in list, otherwise \${idx} will be empty
- type video, data will be converted to encode video binary and write to oss file,

**:: return** { 'image': 'image', 'feature': 'json'

}

indicating that the image data in the output dict will be save to image file and feature in output dict will be converted to json

**batch**(*image\_tensor\_list*)

**predict**(*input\_data\_list*, *batch\_size*=- 1)

using session run predict a number of samples using batch\_size

### Parameters

- **input\_data\_list** – a list of numpy array, each array is a sample to be predicted
- **batch\_size** – batch\_size passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust batch\_size in runtime

### Returns

a list of dict, each dict is the prediction result of one sample eg, {"output1": value1, "output2": value2}, the value type can be python int str float, and numpy array

**Return type** result

## 14.5 easycv.predictors.detector module

**class** easycv.predictors.detector.TorchYoloXPredictor(*model\_path*, *max\_det*=100, *score\_thresh*=0.5, *model\_config*=None)

Bases: [easycv.predictors.interface.PredictorInterface](#)

**\_\_init\_\_**(*model\_path*, *max\_det*=100, *score\_thresh*=0.5, *model\_config*=None)  
init model

### Parameters

- **model\_path** – model file path
- **max\_det** – maximum number of detection
- **score\_thresh** – score\_thresh to filter box
- **model\_config** – config string for model to init, in json format

**predict**(*input\_data\_list*, *batch\_size*=-1)  
using session run predict a number of samples using batch\_size

### Parameters

- **input\_data\_list** – a list of numpy array(in rgb order), each array is a sample to be predicted
- **batch\_size** – batch\_size passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust batch\_size in runtime

### Returns

a list of dict, each dict is the prediction result of one sample eg, {"output1": value1, "output2": value2}, the value type can be python int str float, and numpy array

**Return type** result

**class** easycv.predictors.detector.TorchFaceDetector(*model\_path*=None, *model\_config*=None)

Bases: [easycv.predictors.interface.PredictorInterface](#)

**\_\_init\_\_**(*model\_path*=None, *model\_config*=None)  
init model, add a facedetect and align for img input.

### Parameters

- **model\_path** – model file path
- **model\_config** – config string for model to init, in json format

**get\_output\_type**()

in this function user should return a type dict, which indicates which type of data should the output of predictor be converted to \* type json, data will be serialized to json str

- type image, data will be converted to encode image binary and write to oss file, whose name is output\_dir/{key}/{input\_filename}\_{idx}.jpg, where input\_filename is the base filename extracted from url, key corresponds to the key in the dict of output\_type, if the type of data indexed by key is a list, idx is the index of element in list, otherwise {idx} will be empty
- type video, data will be converted to encode video binary and write to oss file,

**:: return** { 'image': 'image', 'feature': 'json' }

} indicating that the image data in the output dict will be save to image file and feature in output dict will be converted to json

**batch**(*image\_tensor\_list*)

**predict**(*input\_data\_list*, *batch\_size=-1*, *threshold=0.95*)

using session run predict a number of samples using *batch\_size*

#### Parameters

- **input\_data\_list** – a list of numpy array, each array is a sample to be predicted
- **batch\_size** – *batch\_size* passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust *batch\_size* in runtime

#### Returns

**a list of dict, each dict is the prediction result of one sample** eg, {"output1": value1, "output2": value2}, the value type can be python int str float, and numpy array

**Return type** result

**Raises if detect !=1 face in a img, then do nothing for this image –**

```
class easycv.predictors.detector.TorchYoloXClassifierPredictor(models_root_dir, max_det=100,
 cls_score_thresh=0.01,
 det_model_config=None,
 cls_model_config=None)
```

Bases: [easycv.predictors.interface.PredictorInterface](#)

```
__init__(models_root_dir, max_det=100, cls_score_thresh=0.01, det_model_config=None,
 cls_model_config=None)
```

init model, add a yolox and classification predictor for img input.

#### Parameters

- **models\_root\_dir** – *models\_root\_dir/detection/.pth* and *models\_root\_dir/classification/.pth*
- **det\_model\_config** – config string for detection model to init, in json format
- **cls\_model\_config** – config string for classification model to init, in json format

**predict**(*input\_data\_list*, *batch\_size=-1*)

using session run predict a number of samples using *batch\_size*

#### Parameters

- **input\_data\_list** – a list of numpy array(in rgb order), each array is a sample to be predicted
- **batch\_size** – *batch\_size* passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust *batch\_size* in runtime

#### Returns

**a list of dict, each dict is the prediction result of one sample** eg, {"output1": value1, "output2": value2}, the value type can be python int str float, and numpy array

**Return type** result



## 14.6 easycv.predictors.feature\_extractor module

**class** easycv.predictors.feature\_extractor.TorchFeatureExtractor(*model\_path*,  
*model\_config=None*)

Bases: *easycv.predictors.interface.PredictorInterface*

**\_\_init\_\_**(*model\_path*, *model\_config=None*)  
init model

### Parameters

- **model\_path** – model file path
- **model\_config** – config string for model to init, in json format

### get\_output\_type()

in this function user should return a type dict, which indicates which type of data should the output of predictor be converted to \* type json, data will be serialized to json str

- type image, data will be converted to encode image binary and write to oss file, whose name is output\_dir/{key}/{input\_filename}\_{idx}.jpg, where input\_filename is the base filename extracted from url, key corresponds to the key in the dict of output\_type, if the type of data indexed by key is a list, idx is the index of element in list, otherwise {idx} will be empty
- type video, data will be converted to encode video binary and write to oss file,

**:: return** { 'image': 'image', 'feature': 'json' }

} indicating that the image data in the output dict will be save to image file and feature in output dict will be converted to json

**batch**(*image\_tensor\_list*)

**predict**(*input\_data\_list*, *batch\_size=-1*)  
using session run predict a number of samples using batch\_size

### Parameters

- **input\_data\_list** – a list of numpy array, each array is a sample to be predicted
- **batch\_size** – batch\_size passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust batch\_size in runtime

### Returns

**a list of dict, each dict is the prediction result of one sample** eg, {“output1”: value1, “output2”: value2}, the value type can be python int str float, and numpy array

**Return type** result

**class** easycv.predictors.feature\_extractor.TorchFaceFeatureExtractor(*model\_path*,  
*model\_config=None*)

Bases: *easycv.predictors.interface.PredictorInterface*

**\_\_init\_\_**(*model\_path*, *model\_config=None*)  
init model, add a facedetect and align for img input.

### Parameters

- **model\_path** – model file path
- **model\_config** – config string for model to init, in json format

**get\_output\_type()**

in this function user should return a type dict, which indicates which type of data should the output of predictor be converted to \* type json, data will be serialized to json str

- type image, data will be converted to encode image binary and write to oss file, whose name is `output_dir/${key}/${input_filename}_${idx}.jpg`, where `input_filename` is the base filename extracted from url, key corresponds to the key in the dict of `output_type`, if the type of data indexed by key is a list, `idx` is the index of element in list, otherwise `${idx}` will be empty
- type video, data will be converted to encode video binary and write to oss file,

```
:: return { 'image': 'image', 'feature': 'json' }
```

} indicating that the image data in the output dict will be save to image file and feature in output dict will be converted to json

**batch(image\_tensor\_list)**

**predict**(input\_data\_list, batch\_size=- 1, detect\_and\_align=True)

using session run predict a number of samples using batch\_size

**Parameters**

- **input\_data\_list** – a list of numpy array or PIL.Image, each array is a sample to be predicted
- **batch\_size** – batch\_size passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust batch\_size in runtime
- **detect\_and\_align** – True to detect and align before feature extractor

**Returns**

a list of dict, each dict is the prediction result of one sample eg, {"output1": value1, "output2": value2}, the value type can be python int str float, and numpy array

**Return type** result

**Raises if detect !=1 face in a img, then do nothing for this image –**

```
class easycv.predictors.feature_extractor.TorchMultiFaceFeatureExtractor(model_path,
 model_config=None)
```

Bases: [easycv.predictors.interface.PredictorInterface](#)

**\_\_init\_\_**(model\_path, model\_config=None)

init model, add a facedetect and align for img input.

**Parameters**

- **model\_path** – model file path
- **model\_config** – config string for model to init, in json format

**get\_output\_type()**

in this function user should return a type dict, which indicates which type of data should the output of predictor be converted to \* type json, data will be serialized to json str

- type image, data will be converted to encode image binary and write to oss file, whose name is `output_dir/${key}/${input_filename}_${idx}.jpg`, where `input_filename` is the base filename extracted from url, key corresponds to the key in the dict of `output_type`, if the type of data indexed by key is a list, `idx` is the index of element in list, otherwise `${idx}` will be empty
- type video, data will be converted to encode video binary and write to oss file,

```
:: return { 'image': 'image', 'feature': 'json'}
```

} indicating that the image data in the output dict will be save to image file and feature in output dict will be converted to json

```
batch(image_tensor_list)
```

```
predict(input_data_list, batch_size=- 1, detect_and_align=True)
```

using session run predict a number of samples using batch\_size

#### Parameters

- **input\_data\_list** – a list of numpy array or PIL.Image, each array is a sample to be predicted
- **batch\_size** – batch\_size passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust batch\_size in runtime
- **detect\_and\_align** – True to detect and align before feature extractor

#### Returns

**a list of dict, each dict is the prediction result of one sample** eg, {"output1": value1, "output2": value2}, the value type can be python int str float, and numpy array

**Return type** result

**Raises if detect !=1 face in a img, then do nothing for this image –**

```
class easycv.predictors.feature_extractor.TorchFaceAttrExtractor(model_path,
 model_config=None,
 face_threshold=0.95,
 attr_method=['distribute_sum',
 'softmax', 'softmax'],
 attr_name=['age', 'gender',
 'emo'])
```

Bases: [easycv.predictors.interface.PredictorInterface](#)

```
__init__(model_path, model_config=None, face_threshold=0.95, attr_method=['distribute_sum', 'softmax',
 'softmax'], attr_name=['age', 'gender', 'emo'])
init model
```

#### Parameters

- **model\_path** – model file path
- **model\_config** – config string for model to init, in json format
- **attr\_method** –
  - softmax: do softmax for feature\_dim 1
  - distribute\_sum: do softmax and prob sum

#### get\_output\_type()

in this function user should return a type dict, which indicates which type of data should the output of predictor be converted to \* type json, data will be serialized to json str

- type image, data will be converted to encode image binary and write to oss file, whose name is output\_dir/\${key}/\${input\_filename}\_\${idx}.jpg, where input\_filename is the base filename extracted from url, key corresponds to the key in the dict of output\_type, if the type of data indexed by key is a list, idx is the index of element in list, otherwise \${idx} will be empty
- type video, data will be converted to encode video binary and write to oss file,

```
:: return { 'image': 'image', 'feature': 'json'}
```

} indicating that the image data in the output dict will be save to image file and feature in output dict will be converted to json

**batch**(*image\_tensor\_list*)

**predict**(*input\_data\_list*, *batch\_size=-1*)

using session run predict a number of samples using batch\_size

#### Parameters

- **input\_data\_list** – a list of numpy array, each array is a sample to be predicted
- **batch\_size** – batch\_size passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust batch\_size in runtime

#### Returns

**a list of dict, each dict is the prediction result of one sample** eg, {"output1": value1, "output2": value2}, the value type can be python int str float, and numpy array

**Return type** result

## 14.7 easycv.predictors.interface module

**class** easycv.predictors.interface.**PredictorInterface**(*model\_path*, *model\_config=None*)

Bases: object

**version** = 1

**\_\_init\_\_**(*model\_path*, *model\_config=None*)

init model

#### Parameters

- **model\_path** – init model from this directory
- **model\_config** – config string for model to init, in json format

**abstract predict**(*input\_data*, *batch\_size*)

using session run predict a number of samples using batch\_size

#### Parameters

- **input\_data** – a list of numpy array, each array is a sample to be predicted
- **batch\_size** – batch\_size passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust batch\_size in runtime

#### Returns

**a list of dict, each dict is the prediction result of one sample** eg, {"output1": value1, "output2": value2}, the value type can be python int str float, and numpy array

**Return type** result

**get\_output\_type**()

in this function user should return a type dict, which indicates which type of data should the output of predictor be converted to \* type json, data will be serialized to json str

- type image, data will be converted to encode image binary and write to oss file, whose name is output\_dir/{key}/{input\_filename}\_{idx}.jpg, where input\_filename is extracted from url, key corresponds to the key in the dict of output\_type, if the type of data indexed by key is a list, idx is the index of element in list, otherwise {idx} will be empty
- type video, data will be converted to encode video binary and write to oss file,

```
:: return { 'image': 'image', 'feature': 'json'}
```

} indicating that the image data in the output dict will be save to image file and feature in output dict will be converted to json

```
class easycv.predictors.interface.PredictorInterfaceV2(model_path, model_config=None)
```

Bases: [easycv.predictors.interface.PredictorInterface](#)

**version** = 2

```
__init__(model_path, model_config=None)
```

init model

#### Parameters

- **model\_path** – init model from this directory
- **model\_config** – config string for model to init, in json format

```
get_output_type()
```

in this function user should return a type dict, which indicates which type of data should the output of predictor be converted to \* type json, data will be serialized to json str

- type image, data will be converted to encode image binary and write to oss file, whose name is output\_dir/{key}/{input\_filename}\_{idx}.jpg, where input\_filename is the base filename extracted from url, key corresponds to the key in the dict of output\_type, if the type of data indexed by key is a list, idx is the index of element in list, otherwise {idx} will be empty
- type video, data will be converted to encode video binary and write to oss file,

```
:: return { 'image': 'image', 'feature': 'json'}
```

} indicating that the image data in the output dict will be save to image file and feature in output dict will be converted to json

```
abstract predict(input_data_dict_list, batch_size)
```

using session run predict a number of samples using batch\_size

#### Parameters

- **input\_data\_dict\_list** – a list of dict, each dict is a sample data to be predicted
- **batch\_size** – batch\_size passed by the caller, you can also ignore this param and use a fixed number if you do not want to adjust batch\_size in runtime

#### Returns

**a list of dict, each dict is the prediction result of one sample** eg, {"output1": value1, "output2": value2}, the value type can be python int str float, and numpy array

**Return type** result

## 14.8 easycv.predictors.pose\_predictor module

**class** easycv.predictors.pose\_predictor.**LoadImage**(*color\_type='color', channel\_order='rgb'*)

Bases: object

A simple pipeline to load image.

**\_\_init\_\_**(*color\_type='color', channel\_order='rgb'*)

Initialize self. See help(type(self)) for accurate signature.

easycv.predictors.pose\_predictor.**rgetattr**(*obj, attr, \*args*)

**class** easycv.predictors.pose\_predictor.**OutputHook**(*module, outputs=None, as\_tensor=False*)

Bases: object

**\_\_init\_\_**(*module, outputs=None, as\_tensor=False*)

Initialize self. See help(type(self)) for accurate signature.

**register**(*module*)

**remove**()

**class** easycv.predictors.pose\_predictor.**TorchPoseTopDownPredictor**(*model\_path,*  
*model\_config=None*)

Bases: [easycv.predictors.interface.PredictorInterface](#)

Inference a single image with a list of bounding boxes.

**\_\_init\_\_**(*model\_path, model\_config=None*)

init model

### Parameters

- **model\_path** – model file path
- **model\_config** – config string for model to init, in json format

**predict**(*input\_data\_list, batch\_size=-1, return\_heatmap=False*)

Inference pose.

### Parameters

- **input\_data\_list** – A list of image infos, like: [   
 {   
     **'img'** (str | np.ndarray, RGB): Image filename or loaded image.   
     **'detection\_results'** (list | np.ndarray): All bounding boxes (with scores), shaped (N, 4) or (N, 5). (left, top, width, height, [score]) where N is number of bounding boxes.   
 }   
 ]   
 • **batch\_size** – batch size   
 • **return\_heatmap** – return heatmap value or not, default false.

### Returns

{ **'pose\_results'**: list of ndarray[NxKx3]: Predicted pose x, y, score **'pose\_heatmap'** (optional): list of heatmap[N, K, H, W]: Model output heatmap   
 }

```
class easycv.predictors.pose_predictor.TorchPoseTopDownPredictorWithDetector(model_path,
 model_config={'detection':
 {'model_type':
 None, 're-
 served_classes':
 [],
 'score_thresh':
 0.0}, 'pose':
 {'bbox_thr':
 0.3, 'format':
 'xywh'}}})
```

Bases: [easycv.predictors.interface.PredictorInterface](#)

```
SUPPORT_DETECTION_PREDICTORS = {'TorchYoloXPredictor': <class
'easycv.predictors.detector.TorchYoloXPredictor'>}
```

```
__init__(model_path, model_config={'detection': {'model_type': None, 'reserved_classes': [],
'score_thresh': 0.0}, 'pose': {'bbox_thr': 0.3, 'format': 'xywh'}})
 init model
```

#### Parameters

- **model\_path** – pose and detection model file path, split with ,, make sure the first is pose model, second is detection model
- **model\_config** – config string for model to init, in json format

```
process_det_results(outputs, input_data_list, reserved_classes=[])
```

```
predict(input_data_list, batch_size=-1, return_heatmap=False)
```

Inference with pose model and detection model.

#### Parameters

- **input\_data\_list** – A list of images(np.ndarray, RGB)
- **batch\_size** – batch size
- **return\_heatmap** – return heatmap value or not, default false.

#### Returns

```
{ 'pose_results': list of ndarray[NxKx3]: Predicted pose x, y, score 'pose_heatmap'
(optional): list of heatmap[N, K, H, W]: Model output heatmap
}
```

```
easycv.predictors.pose_predictor.vis_pose_result(model, img, result, radius=4, thickness=1,
 kpt_score_thr=0.3, bbox_color='green',
 dataset_info=None, show=False, out_file=None)
```

Visualize the detection results on the image.

#### Parameters

- **model** (*nn.Module*) – The loaded detector.
- **img** (*str* | *np.ndarray*) – Image filename or loaded image.
- **result** (*list[dict]*) – The results to draw over *img* (bbox\_result, pose\_result).
- **radius** (*int*) – Radius of circles.
- **thickness** (*int*) – Thickness of lines.
- **kpt\_score\_thr** (*float*) – The threshold to visualize the keypoints.

- **skeleton** (*list[tuple()]*) – Default None.
- **show** (*bool*) – Whether to show the image. Default True.
- **out\_file** (*str/None*) – The filename of the output visualization image.



## EASYCV.CORE PACKAGE

### 15.1 Subpackages

#### 15.1.1 easycv.core.evaluation package

##### Subpackages

##### easycv.core.evaluation.custom\_cocotools package

##### Submodules

##### easycv.core.evaluation.custom\_cocotools.cocoeval module

**class** easycv.core.evaluation.custom\_cocotools.cocoeval.COCOeval(*cocoGt=None, cocoDt=None, iouType='segm', sigmas=None*)

Bases: object

**\_\_init\_\_**(*cocoGt=None, cocoDt=None, iouType='segm', sigmas=None*)

Initialize CocoEval using coco APIs for gt and dt :param cocoGt: coco object with ground truth annotations :param cocoDt: coco object with detection results :param iouType: type of iou to be computed, bbox for detection task,

segm for segmentation task

**Parameters** **sigmas** – keypoint labelling sigmas.

**Returns** None

**evaluate**()

Run per image evaluation on given images and store results (a list of dict) in self.evalImgs :returns: None

**computeIoU**(*imgId, catId*)

**computeOks**(*imgId, catId*)

**evaluateImg**(*imgId, catId, aRng, maxDet*)

perform evaluation for single category and image :param imgId: image id, string :param catId: category id, string :param aRng: area range, tuple :param maxDet: maximum detection number

**Returns** dict (single image results)

**accumulate**(*p=None*)

Accumulate per image evaluation results and store the result in self.eval :param param p: input params for evaluation

**Returns** None

**summarize()**

Compute and display summary metrics for evaluation results. Note this function can *only* be applied on the default parameter setting

**summarize\_per\_category()**

Compute and display summary metrics for evaluation results *per category*. Note this function can *only* be applied on the default parameter setting

**filter\_annotations()**(*annotations, catIds*)

**makeplot()**(*recThrs, precisions, name, save\_dir=None*)

**analyze()**

Analyze errors

**class** easycv.core.evaluation.custom\_cocotools.cocoeval.Params(*iouType='segm'*)

Bases: object

Params for coco evaluation api

**setDetParams()**

**setKpParams()**

**\_\_init\_\_**(*iouType='segm'*)

Initialize self. See help(type(self)) for accurate signature.

## Submodules

### easycv.core.evaluation.ap module

easycv.core.evaluation.ap.**ap\_per\_class**(*tp, conf, pred\_cls, target\_cls, plot=False, fname='precision-recall\_curve.png'*)

Compute the average precision, given the recall and precision curves. Source: <https://github.com/rafaelpadilla/Object-Detection-Metrics>.

#### Parameters

- **tp** – True positives (ndarray, nx1 or nx10).
- **conf** – Objectness value from 0-1 (ndarray).
- **pred\_cls** – Predicted object classes (ndarray).
- **target\_cls** – True object classes (ndarray).
- **plot** – Plot precision-recall curve at **mAP@0.5**
- **fname** – Plot filename

**Returns** The average precision as computed in py-faster-rcnn.

easycv.core.evaluation.ap.**compute\_ap**(*recall, precision*)

Compute the average precision, given the recall and precision curves. Source: <https://github.com/rbgirshick/py-faster-rcnn>.

#### Parameters

- **recall** – The recall curve (list).
- **precision** – The precision curve (list).

**Returns** The average precision as computed in py-faster-rcnn.

### easycv.core.evaluation.auc\_eval module

```
class easycv.core.evaluation.auc_eval.AucEvaluator(dataset_name=None,
 metric_names=['neck_auc'], neck_num=None)
```

Bases: [easycv.core.evaluation.base\\_evaluator.Evaluator](#)

AUC evaluator for binary classification only.

```
__init__(dataset_name=None, metric_names=['neck_auc'], neck_num=None)
```

#### Parameters

- **dataset\_name** – eval dataset name
- **metric\_names** – eval metrics name
- **neck\_num** – some model contains multi-neck to support multitask, neck\_num means use the no.neck\_num neck output of model to eval

### easycv.core.evaluation.base\_evaluator module

```
class easycv.core.evaluation.base_evaluator.Evaluator(dataset_name=None, metric_names=[])
```

Bases: object

Evaluator interface

```
__init__(dataset_name=None, metric_names=[])
```

Construct eval ops from tensor

#### Parameters

- **dataset\_name** (*str*) – dataset name to be evaluated
- **metric\_names** (*List[str]*) – metric names this evaluator will return

```
evaluate(prediction_dict, groundtruth_dict, **kwargs)
```

property **metric\_names**

### easycv.core.evaluation.builder module

```
easycv.core.evaluation.builder.build_evaluator(evaluator_cfg_list)
```

build evaluator according to metric name

**Parameters** **evaluator\_cfg\_list** – list of evaluator config dict

**Returns** return a list of evaluator

### easycv.core.evaluation.classification\_eval module

```
class easycv.core.evaluation.classification_eval.ClsEvaluator(topk=(1, 5), dataset_name=None,
 metric_names=['neck_top1'],
 neck_num=None)
```

Bases: [easycv.core.evaluation.base\\_evaluator.Evaluator](#)

Classification evaluator.

```
__init__(topk=(1, 5), dataset_name=None, metric_names=['neck_top1'], neck_num=None)
```

**Parameters**

- **top\_k** – tuple of int, evaluate top\_k acc
- **dataset\_name** – eval dataset name
- **metric\_names** – eval metrics name
- **neck\_num** – some model contains multi-neck to support multitask, neck\_num means use the no.neck\_num neck output of model to eval

**easycv.core.evaluation.coco\_evaluation module**

Class for evaluating object detections with COCO metrics.

```
class easycv.core.evaluation.coco_evaluation.CocoDetectionEvaluator(classes, include_metrics_per_category=False, all_metrics_per_category=False, coco_analyze=False, dataset_name=None, metric_names=['DetectionBoxes_Precision/mAP'])
```

Bases: `easycv.core.evaluation.base_evaluator.Evaluator`

Class to evaluate COCO detection metrics.

```
__init__(classes, include_metrics_per_category=False, all_metrics_per_category=False, coco_analyze=False, dataset_name=None, metric_names=['DetectionBoxes_Precision/mAP'])
```

Constructor.

**Parameters**

- **classes** – a list of class name
- **include\_metrics\_per\_category** – If True, include metrics for each category.
- **all\_metrics\_per\_category** – Whether to include all the summary metrics for each category in per\_category\_ap. Be careful with setting it to true if you have more than handful of , because it will pollute your mldash.
- **coco\_analyze** – If True, will analyze the detection result using coco analysis.
- **dataset\_name** – If not None, dataset\_name will be inserted to each metric name.

**clear()**

Clears the state to prepare for a fresh evaluation.

**add\_single\_ground\_truth\_image\_info(image\_id, groundtruth\_dict)**

Adds groundtruth for a single image to be used for evaluation.

If the image has already been added, a warning is logged, and groundtruth is ignored.

**Parameters**

- **image\_id** – A unique string/integer identifier for the image.
- **groundtruth\_dict** – A dictionary containing

**InputDataFields.groundtruth\_boxes** float32 numpy array of shape [num\_boxes, 4] containing *num\_boxes* groundtruth boxes of the format [ymin, xmin, ymax, xmax] in absolute image coordinates.

**InputDataFields.groundtruth\_classes** integer numpy array of shape [num\_boxes] containing 1-indexed groundtruth classes for the boxes. **InputDataFields.groundtruth\_is\_crowd** (optional): integer numpy array of shape [num\_boxes] containing iscrowd flag for groundtruth boxes.

**add\_single\_detected\_image\_info**(*image\_id*, *detections\_dict*)

Adds detections for a single image to be used for evaluation.

If a detection has already been added for this image id, a warning is logged, and the detection is skipped.

#### Parameters

- **image\_id** – A unique string/integer identifier for the image.
- **detections\_dict** – A dictionary containing

**DetectionResultFields.detection\_boxes** float32 numpy array of shape [num\_boxes, 4] containing *num\_boxes* detection boxes of the format [ymin, xmin, ymax, xmax] in absolute image coordinates.

**DetectionResultFields.detection\_scores** float32 numpy array of shape [num\_boxes] containing detection scores for the boxes.

**DetectionResultFields.detection\_classes** integer numpy array of shape [num\_boxes] containing 1-indexed detection classes for the boxes.

**Raises ValueError** – If groundtruth for the image\_id is not available.

```
class easycv.core.evaluation.coco_evaluation.CocoMaskEvaluator(classes, include_metrics_per_category=False,
 dataset_name=None, metric_names=['DetectionMasks_Precision/mAP'])
```

Bases: [easycv.core.evaluation.base\\_evaluator.Evaluator](#)

Class to evaluate COCO detection metrics.

```
__init__(classes, include_metrics_per_category=False, dataset_name=None,
 metric_names=['DetectionMasks_Precision/mAP'])
```

Constructor.

#### Parameters

- **categories** – A list of dicts, each of which has the following keys :id: (required) an integer id uniquely identifying this category. :name: (required) string representing category name e.g., 'cat', 'dog'.
- **include\_metrics\_per\_category** – If True, include metrics for each category.

**clear()**

Clears the state to prepare for a fresh evaluation.

**add\_single\_ground\_truth\_image\_info**(*image\_id*, *groundtruth\_dict*)

Adds groundtruth for a single image to be used for evaluation.

If the image has already been added, a warning is logged, and groundtruth is ignored.

#### Parameters

- **image\_id** – A unique string/integer identifier for the image.
- **groundtruth\_dict** – A dictionary containing :InputDataFields.groundtruth\_boxes: float32 numpy array of shape [num\_boxes, 4] containing *num\_boxes* groundtruth boxes of the format [ymin, xmin, ymax, xmax] in absolute image coordinates.

**InputDataFields.groundtruth\_classes** integer numpy array of shape [num\_boxes] containing 1-indexed groundtruth classes for the boxes.

**InputDataFields.groundtruth\_instance\_masks** uint8 numpy array of shape [num\_boxes, image\_height, image\_width] containing groundtruth masks corresponding to the boxes. The elements of the array must be in {0, 1}.

**add\_single\_detected\_image\_info**(*image\_id*, *detections\_dict*)

Adds detections for a single image to be used for evaluation.

If a detection has already been added for this image id, a warning is logged, and the detection is skipped.

#### Parameters

- **image\_id** – A unique string/integer identifier for the image.
- **detections\_dict** – A dictionary containing - **DetectionResultFields.detection\_scores**: float32 numpy array of shape [num\_boxes] containing detection scores for the boxes. **DetectionResultFields.detection\_classes**: integer numpy array of shape [num\_boxes] containing 1-indexed detection classes for the boxes. **DetectionResultFields.detection\_masks**: optional uint8 numpy array of shape [num\_boxes, image\_height, image\_width] containing instance masks corresponding to the boxes. The elements of the array must be in {0, 1}.

**Raises ValueError** – If groundtruth for the image\_id is not available or if spatial shapes of groundtruth\_instance\_masks and detection\_masks are incompatible.

**class** `easycv.core.evaluation.coco_evaluation.CoCoPoseTopDownEvaluator`(*dataset\_name=None*,  
*metric\_names=['AP']*,  
*\*\*kwargs*)

Bases: `easycv.core.evaluation.base_evaluator.Evaluator`

Class to evaluate COCO keypoint topdown metrics.

**\_\_init\_\_**(*dataset\_name=None*, *metric\_names=['AP']*, *\*\*kwargs*)  
Construct eval ops from tensor

#### Parameters

- **dataset\_name** (*str*) – dataset name to be evaluated
- **metric\_names** (*List[str]*) – metric names this evaluator will return

### `easycv.core.evaluation.coco_tools` module

Wrappers for third party pycocotools to be used within object\_detection.

Note that nothing in this file is tensorflow related and thus cannot be called directly as a slim metric, for example.

TODO(jonathanhuang): wrap as a slim metric in metrics.py

Usage example: given a set of images with ids in the list *image\_ids* and corresponding lists of numpy arrays encoding groundtruth (boxes and classes) and detections (boxes, scores and classes), where elements of each list correspond to detections/annotations of a single image, then evaluation (in multi-class mode) can be invoked as follows:

```
groundtruth_dict = coco_tools.ExportGroundtruthToCOCO(image_ids, groundtruth_boxes_list,
 groundtruth_classes_list, max_num_classes, output_path=None)
```

```
detections_list = coco_tools.ExportDetectionsToCOCO(image_ids, detection_boxes_list, detection_scores_list,
 detection_classes_list, output_path=None)
```

```

groundtruth = coco_tools.COCOWrapper(groundtruth_dict) detections =
groundtruth.LoadAnnotations(detections_list) evaluator = coco_tools.COCOEvalWrapper(groundtruth,
detections,

 agnostic_mode=False)

metrics = evaluator.ComputeMetrics()

```

**class** `easycv.core.evaluation.coco_tools.COCOWrapper`(*dataset*, *detection\_type*='bbox')

Bases: `xtcocotools.coco.COCO`

Wrapper for the pycocotools COCO class.

**\_\_init\_\_**(*dataset*, *detection\_type*='bbox')

COCOWrapper constructor.

See <http://mscoco.org/dataset/#format> for a description of the format. By default, the `coco.COCO` class constructor reads from a JSON file. This function duplicates the same behavior but loads from a dictionary, allowing us to perform evaluation without writing to external storage.

#### Parameters

- **dataset** – a dictionary holding bounding box annotations in the COCO format.
- **detection\_type** – type of detections being wrapped. Can be one of ['bbox', 'segmentation']

**Raises ValueError** – if *detection\_type* is unsupported.

**LoadAnnotations**(*annotations*)

Load annotations dictionary into COCO datastructure.

See <http://mscoco.org/dataset/#format> for a description of the annotations format. As above, this function replicates the default behavior of the API but does not require writing to external storage.

**Parameters annotations** – python list holding object detection results where each detection is encoded as a dict with required keys ['image\_id', 'category\_id', 'score'] and one of ['bbox', 'segmentation'] based on *detection\_type*.

**Returns** a `coco.COCO` datastructure holding object detection annotations results

#### Raises

- **ValueError** – if *annotations* is not a list
- **ValueError** – if *annotations* do not correspond to the images contained in self.

**class** `easycv.core.evaluation.coco_tools.COCOEvalWrapper`(*groundtruth*=None, *detections*=None, *agnostic\_mode*=False, *iou\_type*='bbox')

Bases: `easycv.core.evaluation.custom_cocotools.cocoeval.COCOeval`

Wrapper for the pycocotools COCOeval class.

To evaluate, create two objects (*groundtruth\_dict* and *detections\_list*) using the conventions listed at <http://mscoco.org/dataset/#format>. Then call evaluation as follows:

```

groundtruth = coco_tools.COCOWrapper(groundtruth_dict) detec-
tions = groundtruth.LoadAnnotations(detections_list) evaluator =
coco_tools.COCOEvalWrapper(groundtruth, detections,

 agnostic_mode=False)

metrics = evaluator.ComputeMetrics()

__init__(groundtruth=None, detections=None, agnostic_mode=False, iou_type='bbox')
COCOEvalWrapper constructor.

```

Note that for the area-based metrics to be meaningful, detection and groundtruth boxes must be in image coordinates measured in pixels.

#### Parameters

- **groundtruth** – a `coco.COCO` (or `coco_tools.COCOWrapper`) object holding groundtruth annotations
- **detections** – a `coco.COCO` (or `coco_tools.COCOWrapper`) object holding detections
- **agnostic\_mode** – boolean (default: `False`). If `True`, evaluation ignores class labels, treating all detections as proposals.
- **iou\_type** – IOU type to use for evaluation. Supports *bbox* or *segm*.

#### **GetCategory**(category\_id)

Fetches dictionary holding category information given category id.

**Parameters** `category_id` – integer id

**Returns** dictionary holding 'id', 'name'.

#### **GetAgnosticMode**()

Returns true if COCO Eval is configured to evaluate in agnostic mode.

#### **GetCategoryIdList**()

Returns list of valid category ids.

#### **ComputeMetrics**(include\_metrics\_per\_category=False, all\_metrics\_per\_category=False)

Computes detection metrics.

#### Parameters

- **include\_metrics\_per\_category** – If `True`, will include metrics per category.
- **all\_metrics\_per\_category** – If `true`, include all the summary metrics for each category in `per_category_ap`. Be careful with setting it to `true` if you have more than handful of categories, because it will pollute your `mldash`.

#### Returns

a dictionary holding:

**'Precision/mAP': mean average precision over classes averaged over IOU**  
thresholds ranging from .5 to .95 with .05 increments

**'Precision/mAP@.50IOU':** mean average precision at 50% IOU **'Precision/mAP@.75IOU':** mean average precision at 75% IOU **'Precision/mAP (small)':** mean average precision for small objects  
(area < 32<sup>2</sup> pixels)

**'Precision/mAP (medium)':** mean average precision for medium sized  
objects (32<sup>2</sup> pixels < area < 96<sup>2</sup> pixels)

**'Precision/mAP (large)':** mean average precision for large objects (96<sup>2</sup> pixels < area < 10000<sup>2</sup> pixels)

**'Recall/AR@1':** average recall with 1 detection **'Recall/AR@10':** average recall with 10 detections **'Recall/AR@100':** average recall with 100 detections **'Recall/AR@100 (small)':** average recall for small objects with 100 detections



**'Recall/AR@100 (medium)'**: average recall for medium objects with 100 detections

**'Recall/AR@100 (large)'**: average recall for large objects with 100 detections

2. `per_category_ap`: a dictionary holding category specific results with

keys of the form: 'Precision mAP ByCategory/category' (without the supercategory part if no supercategories exist). For backward compatibility 'PerformanceByCategory' is included in the output regardless of `all_metrics_per_category`. If evaluating class-agnostic mode, `per_category_ap` is an empty dictionary.

### Return type

1. `summary_metrics`

**Raises ValueError** – If `category_stats` does not exist.

### Analyze()

Analyze detection results.

Args:

### Returns

A dictionary containing images of analyzing result images, key is the image name, value is a [H,W,3] numpy array which represent the image content. You can refer to <http://cocodataset.org/#detection-eval> section 4 Analysis code.

```
easycv.core.evaluation.coco_tools.ExportSingleImageGroundtruthToCoco(image_id,
 next_annotation_id,
 category_id_set,
 groundtruth_boxes,
 groundtruth_classes,
 groundtruth_masks=None,
 groundtruth_is_crowd=None,
 super_categories=None)
```

Export groundtruth of a single image to COCO format.

This function converts groundtruth detection annotations represented as numpy arrays to dictionaries that can be ingested by the COCO evaluation API. Note that the `image_ids` provided here must match the ones given to `ExportSingleImageDetectionsToCoco`. We assume that `boxes` and `classes` are in correspondence - that is: `groundtruth_boxes[i, :]`, and `groundtruth_classes[i]` are associated with the same groundtruth annotation.

In the exported result, "area" fields are always set to the area of the groundtruth bounding box.

### Parameters

- **image\_id** – a unique image identifier either of type integer or string.
- **next\_annotation\_id** – integer specifying the first id to use for the groundtruth annotations. All annotations are assigned a continuous integer id starting from this value.
- **category\_id\_set** – A set of valid class ids. Groundtruth with classes not in `category_id_set` are dropped.
- **groundtruth\_boxes** – numpy array (float32) with shape `[num_gt_boxes, 4]`
- **groundtruth\_classes** – numpy array (int) with shape `[num_gt_boxes]`
- **groundtruth\_masks** – optional uint8 numpy array of shape `[num_detections, image_height, image_width]` containing detection\_masks.

- **groundtruth\_is\_crowd** – optional numpy array (int) with shape [num\_gt\_boxes] indicating whether groundtruth boxes are crowd.
- **super\_categories** – optional list of str indicating each box super category

**Returns** a list of groundtruth annotations for a single image in the COCO format.

**Raises ValueError** – if (1) groundtruth\_boxes and groundtruth\_classes do not have the right lengths or (2) if each of the elements inside these lists do not have the correct shapes or (3) if image\_ids are not integers

```
easycv.core.evaluation.coco_tools.ExportGroundtruthToCOCO(image_ids, groundtruth_boxes,
 groundtruth_classes, categories,
 output_path=None)
```

Export groundtruth detection annotations in numpy arrays to COCO API.

This function converts a set of groundtruth detection annotations represented as numpy arrays to dictionaries that can be ingested by the COCO API. Inputs to this function are three lists: image ids for each groundtruth image, groundtruth boxes for each image and groundtruth classes respectively. Note that the image\_ids provided here must match the ones given to the ExportDetectionsToCOCO function in order for evaluation to work properly. We assume that for each image, boxes, scores and classes are in correspondence — that is: image\_id[i], groundtruth\_boxes[i, :] and groundtruth\_classes[i] are associated with the same groundtruth annotation.

In the exported result, “area” fields are always set to the area of the groundtruth bounding box and “iscrowd” fields are always set to 0. TODO(jonathanhuang): pass in “iscrowd” array for evaluating on COCO dataset.

#### Parameters

- **image\_ids** – a list of unique image identifier either of type integer or string.
- **groundtruth\_boxes** – list of numpy arrays with shape [num\_gt\_boxes, 4] (note that num\_gt\_boxes can be different for each entry in the list)
- **groundtruth\_classes** – list of numpy arrays (int) with shape [num\_gt\_boxes] (note that num\_gt\_boxes can be different for each entry in the list)
- **categories** – a list of dictionaries representing all possible categories. Each dict in this list has the following keys:

‘id’: (required) an integer id uniquely identifying this category ‘name’: (required) string representing category name

e.g., ‘cat’, ‘dog’, ‘pizza’

‘supercategory’: (optional) string representing the supercategory e.g., ‘animal’, ‘vehicle’, ‘food’, etc

- **output\_path** – (optional) path for exporting result to JSON

**Returns** dictionary that can be read by COCO API

**Raises ValueError** – if (1) groundtruth\_boxes and groundtruth\_classes do not have the right lengths or (2) if each of the elements inside these lists do not have the correct shapes or (3) if image\_ids are not integers

```
easycv.core.evaluation.coco_tools.ExportSingleImageDetectionBoxesToCoco(image_id,
 category_id_set,
 detection_boxes,
 detection_scores,
 detection_classes)
```

Export detections of a single image to COCO format.

This function converts detections represented as numpy arrays to dictionaries that can be ingested by the COCO evaluation API. Note that the `image_ids` provided here must match the ones given to the `ExportSingleImageDetectionBoxesToCoco`. We assume that boxes, and classes are in correspondence - that is: `boxes[i, :]`, and `classes[i]` are associated with the same groundtruth annotation.

#### Parameters

- **image\_id** – unique image identifier either of type integer or string.
- **category\_id\_set** – A set of valid class ids. Detections with classes not in `category_id_set` are dropped.
- **detection\_boxes** – float numpy array of shape `[num_detections, 4]` containing detection boxes.
- **detection\_scores** – float numpy array of shape `[num_detections]` containing scored for the detection boxes.
- **detection\_classes** – integer numpy array of shape `[num_detections]` containing the classes for detection boxes.

**Returns** a list of detection annotations for a single image in the COCO format.

**Raises ValueError** – if (1) `detection_boxes`, `detection_scores` and `detection_classes` do not have the right lengths or (2) if each of the elements inside these lists do not have the correct shapes or (3) if `image_ids` are not integers.

```
easycv.core.evaluation.coco_tools.ExportSingleImageDetectionMasksToCoco(image_id,
 category_id_set,
 detection_masks,
 detection_scores,
 detection_classes)
```

Export detection masks of a single image to COCO format.

This function converts detections represented as numpy arrays to dictionaries that can be ingested by the COCO evaluation API. We assume that `detection_masks`, `detection_scores`, and `detection_classes` are in correspondence - that is: `detection_masks[i, :]`, `detection_classes[i]` and `detection_scores[i]` are associated with the same annotation.

#### Parameters

- **image\_id** – unique image identifier either of type integer or string.
- **category\_id\_set** – A set of valid class ids. Detections with classes not in `category_id_set` are dropped.
- **detection\_masks** – uint8 numpy array of shape `[num_detections, image_height, image_width]` containing detection\_masks.
- **detection\_scores** – float numpy array of shape `[num_detections]` containing scores for detection masks.
- **detection\_classes** – integer numpy array of shape `[num_detections]` containing the classes for detection masks.

**Returns** a list of detection mask annotations for a single image in the COCO format.

**Raises ValueError** – if (1) `detection_masks`, `detection_scores` and `detection_classes` do not have the right lengths or (2) if each of the elements inside these lists do not have the correct shapes or (3) if `image_ids` are not integers.

```
easycv.core.evaluation.coco_tools.ExportDetectionsToCOCO(image_ids, detection_boxes,
 detection_scores, detection_classes,
 categories, output_path=None)
```

Export detection annotations in numpy arrays to COCO API.

This function converts a set of predicted detections represented as numpy arrays to dictionaries that can be ingested by the COCO API. Inputs to this function are lists, consisting of boxes, scores and classes, respectively, corresponding to each image for which detections have been produced. Note that the `image_ids` provided here must match the ones given to the `ExportGroundtruthToCOCO` function in order for evaluation to work properly.

We assume that for each image, boxes, scores and classes are in correspondence — that is: `detection_boxes[i, :]`, `detection_scores[i]` and `detection_classes[i]` are associated with the same detection.

#### Parameters

- **image\_ids** – a list of unique image identifier either of type integer or string.
- **detection\_boxes** – list of numpy arrays with shape `[num_detection_boxes, 4]`
- **detection\_scores** – list of numpy arrays (float) with shape `[num_detection_boxes]`. Note that `num_detection_boxes` can be different for each entry in the list.
- **detection\_classes** – list of numpy arrays (int) with shape `[num_detection_boxes]`. Note that `num_detection_boxes` can be different for each entry in the list.
- **categories** – a list of dictionaries representing all possible categories. Each dict in this list must have an integer 'id' key uniquely identifying this category.
- **output\_path** – (optional) path for exporting result to JSON

**Returns** list of dictionaries that can be read by COCO API, where each entry corresponds to a single detection and has keys from: `['image_id', 'category_id', 'bbox', 'score']`.

**Raises ValueError** – if (1) `detection_boxes` and `detection_classes` do not have the right lengths or (2) if each of the elements inside these lists do not have the correct shapes or (3) if `image_ids` are not integers.

```
easycv.core.evaluation.coco_tools.ExportSegmentsToCOCO(image_ids, detection_masks,
 detection_scores, detection_classes,
 categories, output_path=None)
```

Export segmentation masks in numpy arrays to COCO API.

This function converts a set of predicted instance masks represented as numpy arrays to dictionaries that can be ingested by the COCO API. Inputs to this function are lists, consisting of segments, scores and classes, respectively, corresponding to each image for which detections have been produced.

Note this function is recommended to use for small dataset. For large dataset, it should be used with a merge function (e.g. in map reduce), otherwise the memory consumption is large.

We assume that for each image, masks, scores and classes are in correspondence — that is: `detection_masks[i, :, :]`, `detection_scores[i]` and `detection_classes[i]` are associated with the same detection.

#### Parameters

- **image\_ids** – list of image ids (typically ints or strings)
- **detection\_masks** – list of numpy arrays with shape `[num_detection, h, w, 1]` and type `uint8`. The height and width should match the shape of corresponding image.
- **detection\_scores** – list of numpy arrays (float) with shape `[num_detection]`. Note that `num_detection` can be different for each entry in the list.
- **detection\_classes** – list of numpy arrays (int) with shape `[num_detection]`. Note that `num_detection` can be different for each entry in the list.

- **categories** – a list of dictionaries representing all possible categories. Each dict in this list must have an integer ‘id’ key uniquely identifying this category.
- **output\_path** – (optional) path for exporting result to JSON

**Returns** list of dictionaries that can be read by COCO API, where each entry corresponds to a single detection and has keys from: [‘image\_id’, ‘category\_id’, ‘segmentation’, ‘score’].

**Raises ValueError** – if detection\_masks and detection\_classes do not have the right lengths or if each of the elements inside these lists do not have the correct shapes.

```
easycv.core.evaluation.coco_tools.ExportKeypointsToCOCO(image_ids, detection_keypoints,
 detection_scores, detection_classes,
 categories, output_path=None)
```

Exports keypoints in numpy arrays to COCO API.

This function converts a set of predicted keypoints represented as numpy arrays to dictionaries that can be ingested by the COCO API. Inputs to this function are lists, consisting of keypoints, scores and classes, respectively, corresponding to each image for which detections have been produced.

We assume that for each image, keypoints, scores and classes are in correspondence — that is: detection\_keypoints[i, :, :], detection\_scores[i] and detection\_classes[i] are associated with the same detection.

#### Parameters

- **image\_ids** – list of image ids (typically ints or strings)
- **detection\_keypoints** – list of numpy arrays with shape [num\_detection, num\_keypoints, 2] and type float32 in absolute x-y coordinates.
- **detection\_scores** – list of numpy arrays (float) with shape [num\_detection]. Note that num\_detection can be different for each entry in the list.
- **detection\_classes** – list of numpy arrays (int) with shape [num\_detection]. Note that num\_detection can be different for each entry in the list.
- **categories** – a list of dictionaries representing all possible categories. Each dict in this list must have an integer ‘id’ key uniquely identifying this category and an integer ‘num\_keypoints’ key specifying the number of keypoints the category has.
- **output\_path** – (optional) path for exporting result to JSON

**Returns** list of dictionaries that can be read by COCO API, where each entry corresponds to a single detection and has keys from: [‘image\_id’, ‘category\_id’, ‘keypoints’, ‘score’].

**Raises ValueError** – if detection\_keypoints and detection\_classes do not have the right lengths or if each of the elements inside these lists do not have the correct shapes.

### easycv.core.evaluation.faceid\_pair\_eval module

```
easycv.core.evaluation.faceid_pair_eval.calculate_roc(thresholds, embeddings1, embeddings2,
 actual_issame, nrof_folds=10, pca=0)
```

```
easycv.core.evaluation.faceid_pair_eval.calculate_accuracy(threshold, dist, actual_issame)
```

```
easycv.core.evaluation.faceid_pair_eval.calculate_val(thresholds, embeddings1, embeddings2,
 actual_issame, far_target, nrof_folds=10)
```

```
easycv.core.evaluation.faceid_pair_eval.calculate_val_far(threshold, dist, actual_issame)
```

`easycv.core.evaluation.faceid_pair_eval.faceid_evaluate(embeddings, actual_issame, nrof_folds=10, pca=0)`

Do Kfold=nrof\_folds faceid pair-match test for embeddings :param embeddings: [N x C] inputs embedding of all dataset :param actual\_issame: [N/2, 1] label of is match :param nrof\_folds: KFold number :param pca: > 0 means, do pca and trans embedding to [N, pca] feature

**Returns** KFold average best accuracy and best threshold

**class** `easycv.core.evaluation.faceid_pair_eval.FaceIDPairEvaluator(dataset_name=None, metric_names=['acc'], kfold=10, pca=0)`

Bases: `easycv.core.evaluation.base_evaluator.Evaluator`

FaceIDPairEvaluator evaluator. Input nx2 pairs and label, kfold thresholds search and return average best accuracy

**\_\_init\_\_**(`dataset_name=None, metric_names=['acc'], kfold=10, pca=0`)

Faceid small dataset evaluator, do pair match validation :param dataset\_name: faceid small validate set name, include [lfw, agedb\_30, cfp\_ff, cfp\_fw, calfw] :param kfold: Kfold for train/val split :param pca: pca dimensions, if > 0, do PCA for input feature, transfer to [n, pca]

**Returns** None

### `easycv.core.evaluation.metric_registry` module

**class** `easycv.core.evaluation.metric_registry.MetricRegistry`

Bases: object

**\_\_init\_\_**()

Initialize self. See help(type(self)) for accurate signature.

**get**(`evaluator_type`)

**register\_default\_best\_metric**(`cls, metric_name, metric_cmp_op='max'`)

Register default best metric for each evaluator

#### Parameters

- **cls** (*object*) – class object
- **metric\_name** (*str or List[str]*) – default best metric name
- **metric\_cmp\_op** (*str or List[str]*) – metric compare operation, should be one of ["max", "min"]

### `easycv.core.evaluation.mse_eval` module

**class** `easycv.core.evaluation.mse_eval.MSEEvaluator(dataset_name=None, metric_names=['avg_mse'], neck_num=None)`

Bases: `easycv.core.evaluation.base_evaluator.Evaluator`

MSEEvaluator evaluator,

**\_\_init\_\_**(`dataset_name=None, metric_names=['avg_mse'], neck_num=None`)

**easycv.core.evaluation.retrival\_topk\_eval module**

```
class easycv.core.evaluation.retrival_topk_eval.RetrievalTopKEvaluator(topk=(1, 2, 4, 8),
 norm=0, metric='cos',
 pca=0,
 dataset_name=None,
 met-
 ric_names=['R@K=1'],
 save_results=False,
 save_results_dir='', fea-
 ture_keyword=['neck'])
```

Bases: `easycv.core.evaluation.base_evaluator.Evaluator`

RetrievalTopK evaluator, Retrieval evaluate do the topK retrieval, by measuring the distance of every 1 vs other. get the topK nearest, and count the match of ID. if Retrieval = 1, Miss = 0. Finally average all RetrievalRate.

```
__init__(topk=(1, 2, 4, 8), norm=0, metric='cos', pca=0, dataset_name=None, metric_names=['R@K=1'],
 save_results=False, save_results_dir='', feature_keyword=['neck'])
```

**Parameters** **top\_k** – tuple of int, evaluate top\_k acc

**easycv.core.evaluation.top\_down\_eval module**

```
easycv.core.evaluation.top_down_eval.pose_pck_accuracy(output, target, mask, thr=0.05,
 normalize=None)
```

Calculate the pose accuracy of PCK for each individual keypoint and the averaged accuracy across all keypoints from heatmaps.

---

**Note:** PCK metric measures accuracy of the localization of the body joints. The distances between predicted positions and the ground-truth ones are typically normalized by the bounding box size. The threshold (thr) of the normalized distance is commonly set as 0.05, 0.1 or 0.2 etc.

batch\_size: N num\_keypoints: K heatmap height: H heatmap width: W

---

**Parameters**

- **output** (`np.ndarray[N, K, H, W]`) – Model output heatmaps.
- **target** (`np.ndarray[N, K, H, W]`) – Groundtruth heatmaps.
- **mask** (`np.ndarray[N, K]`) – Visibility of the target. False for invisible joints, and True for visible. Invisible joints will be ignored for accuracy calculation.
- **thr** (`float`) – Threshold of PCK calculation. Default 0.05.
- **normalize** (`np.ndarray[N, 2]`) – Normalization factor for H&W.

**Returns**

A tuple containing keypoint accuracy.

- `np.ndarray[K]`: Accuracy of each keypoint.
- `float`: Averaged accuracy across all keypoints.
- `int`: Number of valid keypoints.

**Return type** tuple

`easycv.core.evaluation.top_down_eval.keypoint_pck_accuracy(pred, gt, mask, thr, normalize)`

Calculate the pose accuracy of PCK for each individual keypoint and the averaged accuracy across all keypoints for coordinates.

---

**Note:** PCK metric measures accuracy of the localization of the body joints. The distances between predicted positions and the ground-truth ones are typically normalized by the bounding box size. The threshold (*thr*) of the normalized distance is commonly set as 0.05, 0.1 or 0.2 etc.

*batch\_size*: N *num\_keypoints*: K

---

#### Parameters

- **pred** (`np.ndarray[N, K, 2]`) – Predicted keypoint location.
- **gt** (`np.ndarray[N, K, 2]`) – Groundtruth keypoint location.
- **mask** (`np.ndarray[N, K]`) – Visibility of the target. False for invisible joints, and True for visible. Invisible joints will be ignored for accuracy calculation.
- **thr** (`float`) – Threshold of PCK calculation.
- **normalize** (`np.ndarray[N, 2]`) – Normalization factor for H&W.

#### Returns

A tuple containing keypoint accuracy.

- **acc** (`np.ndarray[K]`): Accuracy of each keypoint.
- **avg\_acc** (`float`): Averaged accuracy across all keypoints.
- **cnt** (`int`): Number of valid keypoints.

#### Return type tuple

`easycv.core.evaluation.top_down_eval.post_dark_udp(coords, batch_heatmaps, kernel=3)`

DARK post-processing. Implemented by udp. Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020). Zhang et al. Distribution-Aware Coordinate Representation for Human Pose Estimation (CVPR 2020).

---

**Note:** *batch size*: B *num keypoints*: K *num persons*: N *height of heatmaps*: H *width of heatmaps*: W B=1 for bottom\_up paradigm where all persons share the same heatmap. B=N for top\_down paradigm where each person has its own heatmaps.

---

#### Parameters

- **coords** (`np.ndarray[N, K, 2]`) – Initial coordinates of human pose.
- **batch\_heatmaps** (`np.ndarray[B, K, H, W]`) – batch\_heatmaps
- **kernel** (`int`) – Gaussian kernel size (K) for modulation.

**Returns** Refined coordinates.

**Return type** `res` (`np.ndarray[N, K, 2]`)



```

easycv.core.evaluation.top_down_eval.keypoints_from_heatmaps(heatmaps, center, scale,
 unbiased=False,
 post_process='default', kernel=11,
 valid_radius_factor=0.0546875,
 use_udp=False,
 target_type='GaussianHeatmap')

```

Get final keypoint predictions from heatmaps and transform them back to the image.

---

**Note:** batch size: N num keypoints: K heatmap height: H heatmap width: W

---

### Parameters

- **heatmaps** (*np.ndarray*[N, K, H, W]) – model predicted heatmaps.
- **center** (*np.ndarray*[N, 2]) – Center of the bounding box (x, y).
- **scale** (*np.ndarray*[N, 2]) – Scale of the bounding box wrt height/width.
- **post\_process** (*str/None*) – Choice of methods to post-process heatmaps. Currently supported: None, 'default', 'unbiased', 'megvii'.
- **unbiased** (*bool*) – Option to use unbiased decoding. Mutually exclusive with megvii. Note: this arg is deprecated and unbiased=True can be replaced by post\_process='unbiased' Paper ref: Zhang et al. Distribution-Aware Coordinate Representation for Human Pose Estimation (CVPR 2020).
- **kernel** (*int*) – Gaussian kernel size (K) for modulation, which should match the heatmap gaussian sigma when training. K=17 for sigma=3 and k=11 for sigma=2.
- **valid\_radius\_factor** (*float*) – The radius factor of the positive area in classification heatmap for UDP.
- **use\_udp** (*bool*) – Use unbiased data processing.
- **target\_type** (*str*) – 'GaussianHeatmap' or 'CombinedTarget'. GaussianHeatmap: Classification target with gaussian distribution. CombinedTarget: The combination of classification target (response map) and regression target (offset map). Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020).

### Returns

A tuple containing keypoint predictions and scores.

- preds (*np.ndarray*[N, K, 2]): Predicted keypoint location in images.
- maxvals (*np.ndarray*[N, K, 1]): Scores (confidence) of the keypoints.

**Return type** tuple

## 15.1.2 easycv.core.optimizer package

### Submodules

#### easycv.core.optimizer.lars module

**class** easycv.core.optimizer.lars.**LARS**(*params*, *lr*=<required parameter>, *momentum*=0, *dampening*=0, *weight\_decay*=0, *eta*=0.001, *nesterov*=False)

Bases: torch.optim.optimizer.Optimizer

Implements layer-wise adaptive rate scaling for SGD.

#### Parameters

- **params** (*iterable*) – iterable of parameters to optimize or dicts defining parameter groups
- **lr** (*float*) – base learning rate (gamma\_0)
- **momentum** (*float*, *optional*) – momentum factor (default: 0) (“m”)
- **weight\_decay** (*float*, *optional*) – weight decay (L2 penalty) (default: 0) (“beta”)
- **dampening** (*float*, *optional*) – dampening for momentum (default: 0)
- **eta** (*float*, *optional*) – LARS coefficient
- **nesterov** (*bool*, *optional*) – enables Nesterov momentum (default: False)

Based on Algorithm 1 of the following paper by You, Gitman, and Ginsburg. Large Batch Training of Convolutional Networks:

<https://arxiv.org/abs/1708.03888>

#### Example

```
>>> optimizer = LARS(model.parameters(), lr=0.1, momentum=0.9,
>>> weight_decay=1e-4, eta=1e-3)
>>> optimizer.zero_grad()
>>> loss_fn(model(input), target).backward()
>>> optimizer.step()
```

**\_\_init\_\_**(*params*, *lr*=<required parameter>, *momentum*=0, *dampening*=0, *weight\_decay*=0, *eta*=0.001, *nesterov*=False)

Initialize self. See help(type(self)) for accurate signature.

**step**(*closure*=None)

Performs a single optimization step.

**Parameters** **closure** (*callable*, *optional*) – A closure that reevaluates the model and returns the loss.

**easycv.core.optimizer.ranger module**

`easycv.core.optimizer.ranger.centralized_gradient(x, use_gc=True, gc_conv_only=False)`  
 credit - <https://github.com/Yonghongwei/Gradient-Centralization>

**class** `easycv.core.optimizer.ranger.Ranger(params, lr=0.001, alpha=0.5, k=6, N_sma_threshhold=5, betas=(0.95, 0.999), eps=1e-05, weight_decay=0, use_gc=True, gc_conv_only=False, gc_loc=True)`

Bases: `torch.optim.optimizer.Optimizer`

Adam+LookAhead: refer to <https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer>

**\_\_init\_\_**(*params, lr=0.001, alpha=0.5, k=6, N\_sma\_threshhold=5, betas=(0.95, 0.999), eps=1e-05, weight\_decay=0, use\_gc=True, gc\_conv\_only=False, gc\_loc=True*)

Initialize self. See help(type(self)) for accurate signature.

**step**(*closure=None*)

Performs a single optimization step (parameter update).

**Parameters** **closure** (*callable*) – A closure that reevaluates the model and returns the loss. Optional for most optimizers.

---

**Note:** Unless otherwise specified, this function should not modify the `.grad` field of the parameters.

---

**15.1.3 easycv.core.post\_processing package**

`easycv.core.post_processing.affine_transform(pt, trans_mat)`

Apply an affine transformation to the points.

**Parameters**

- **pt** (*np.ndarray*) – a 2 dimensional point to be transformed
- **trans\_mat** (*np.ndarray*) – 2x3 matrix of an affine transform

**Returns** Transformed points.

**Return type** `np.ndarray`

`easycv.core.post_processing.flip_back(output_flipped, flip_pairs, target_type='GaussianHeatmap')`

Flip the flipped heatmaps back to the original form.

---

**Note:** batch\_size: N num\_keypoints: K heatmap height: H heatmap width: W

---

**Parameters**

- **output\_flipped** (*np.ndarray[N, K, H, W]*) – The output heatmaps obtained from the flipped images.
- **flip\_pairs** (*list[tuple()]*) – Pairs of keypoints which are mirrored (for example, left ear – right ear).
- **target\_type** (*str*) – GaussianHeatmap or CombinedTarget

**Returns** heatmaps that flipped back to the original image

**Return type** `np.ndarray`

`easycv.core.post_processing.fliplr_joints(joints_3d, joints_3d_visible, img_width, flip_pairs)`  
Flip human joints horizontally.

---

**Note:** num\_keypoints: K

---

#### Parameters

- **joints\_3d** (`np.ndarray([K, 3])`) – Coordinates of keypoints.
- **joints\_3d\_visible** (`np.ndarray([K, 1])`) – Visibility of keypoints.
- **img\_width** (`int`) – Image width.
- **flip\_pairs** (`list[tuple()]`) – Pairs of keypoints which are mirrored (for example, left ear – right ear).

#### Returns

Flipped human joints.

- **joints\_3d\_flipped** (`np.ndarray([K, 3])`): Flipped joints.
- **joints\_3d\_visible\_flipped** (`np.ndarray([K, 1])`): Joint visibility.

#### Return type

`easycv.core.post_processing.fliplr_regression(regression, flip_pairs, center_mode='static', center_x=0.5, center_index=0)`

Flip human joints horizontally.

---

**Note:** batch\_size: N num\_keypoint: K

---

#### Parameters

- **regression** (`np.ndarray([..., K, C])`) – Coordinates of keypoints, where K is the joint number and C is the dimension. Example shapes are: - [N, K, C]: a batch of keypoints where N is the batch size. - [N, T, K, C]: a batch of pose sequences, where T is the frame number.
- **flip\_pairs** (`list[tuple()]`) – Pairs of keypoints which are mirrored (for example, left ear – right ear).
- **center\_mode** (`str`) – The mode to set the center location on the x-axis to flip around. Options are: - static: use a static x value (see center\_x also) - root: use a root joint (see center\_index also)
- **center\_x** (`float`) – Set the x-axis location of the flip center. Only used when center\_mode=static.
- **center\_index** (`int`) – Set the index of the root joint, whose x location will be used as the flip center. Only used when center\_mode=root.

#### Returns

Flipped human joints.

- **regression\_flipped** (`np.ndarray([..., K, C])`): Flipped joints.

#### Return type

`easycv.core.post_processing.get_affine_transform(center, scale, rot, output_size, shift=(0.0, 0.0), inv=False)`

Get the affine transform matrix, given the center/scale/rot/output\_size.

**Parameters**

- **center** (`np.ndarray[2, ]`) – Center of the bounding box (x, y).
- **scale** (`np.ndarray[2, ]`) – Scale of the bounding box wrt [width, height].
- **rot** (`float`) – Rotation angle (degree).
- **output\_size** (`np.ndarray[2, ]` | `list(2,)`) – Size of the destination heatmaps.
- **shift** (`0-100%`) – Shift translation ratio wrt the width/height. Default (0., 0.).
- **inv** (`bool`) – Option to inverse the affine transform direction. (inv=False: src->dst or inv=True: dst->src)

**Returns** The transform matrix.

**Return type** `np.ndarray`

`easycv.core.post_processing.get_warp_matrix(theta, size_input, size_dst, size_target)`

Calculate the transformation matrix under the constraint of unbiased. Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020).

**Parameters**

- **theta** (`float`) – Rotation angle in degrees.
- **size\_input** (`np.ndarray`) – Size of input image [w, h].
- **size\_dst** (`np.ndarray`) – Size of output image [w, h].
- **size\_target** (`np.ndarray`) – Size of ROI in input plane [w, h].

**Returns** A matrix for transformation.

**Return type** `matrix (np.ndarray)`

`easycv.core.post_processing.rotate_point(pt, angle_rad)`

Rotate a point by an angle.

**Parameters**

- **pt** (`list[float]`) – 2 dimensional point to be rotated
- **angle\_rad** (`float`) – rotation angle by radian

**Returns** Rotated point.

**Return type** `list[float]`

`easycv.core.post_processing.transform_preds(coords, center, scale, output_size, use_udp=False)`

Get final keypoint predictions from heatmaps and apply scaling and translation to map them back to the image.

---

**Note:** num\_keypoints: K

---

**Parameters**

- **coords** (`np.ndarray[K, ndims]`) –
  - If ndims=2, coords are predicted keypoint location.
  - If ndims=4, coords are composed of (x, y, scores, tags)
  - If ndims=5, coords are composed of (x, y, scores, tags, flipped\_tags)

- **center** (*np.ndarray*[2, ]) – Center of the bounding box (x, y).
- **scale** (*np.ndarray*[2, ]) – Scale of the bounding box wrt [width, height].
- **output\_size** (*np.ndarray*[2, ] | *list*(2,)) – Size of the destination heatmaps.
- **use\_udp** (*bool*) – Use unbiased data processing

**Returns** Predicted coordinates in the images.

**Return type** *np.ndarray*

`easycv.core.post_processing.warp_affine_joints(joints, mat)`

Apply affine transformation defined by the transform matrix on the joints.

**Parameters**

- **joints** (*np.ndarray*[..., 2]) – Origin coordinate of joints.
- **mat** (*np.ndarray*[3, 2]) – The affine matrix.

**Returns** Result coordinate of joints.

**Return type** *matrix* (*np.ndarray*[..., 2])

`easycv.core.post_processing.oks_nms(kpts_db, thr, sigmas=None, vis_thr=None)`

OKS NMS implementations.

**Parameters**

- **kpts\_db** – keypoints.
- **thr** – Retain overlap < thr.
- **sigmas** – standard deviation of keypoint labelling.
- **vis\_thr** – threshold of the keypoint visibility.

**Returns** indexes to keep.

**Return type** *np.ndarray*

`easycv.core.post_processing.soft_oks_nms(kpts_db, thr, max_dets=20, sigmas=None, vis_thr=None)`

Soft OKS NMS implementations.

**Parameters**

- **kpts\_db** –
- **thr** – retain oks overlap < thr.
- **max\_dets** – max number of detections to keep.
- **sigmas** – Keypoint labelling uncertainty.

**Returns** indexes to keep.

**Return type** *np.ndarray*

## Submodules

### easycv.core.post\_processing.nms module

`easycv.core.post_processing.nms.oks_iou(g, d, a_g, a_d, sigmas=None, vis_thr=None)`

Calculate oks ious.

#### Parameters

- **g** – Ground truth keypoints.
- **d** – Detected keypoints.
- **a\_g** – Area of the ground truth object.
- **a\_d** – Area of the detected object.
- **sigmas** – standard deviation of keypoint labelling.
- **vis\_thr** – threshold of the keypoint visibility.

**Returns** The oks ious.

**Return type** list

`easycv.core.post_processing.nms.oks_nms(kpts_db, thr, sigmas=None, vis_thr=None)`

OKS NMS implementations.

#### Parameters

- **kpts\_db** – keypoints.
- **thr** – Retain overlap < thr.
- **sigmas** – standard deviation of keypoint labelling.
- **vis\_thr** – threshold of the keypoint visibility.

**Returns** indexes to keep.

**Return type** np.ndarray

`easycv.core.post_processing.nms.soft_oks_nms(kpts_db, thr, max_dets=20, sigmas=None, vis_thr=None)`

Soft OKS NMS implementations.

#### Parameters

- **kpts\_db** –
- **thr** – retain oks overlap < thr.
- **max\_dets** – max number of detections to keep.
- **sigmas** – Keypoint labelling uncertainty.

**Returns** indexes to keep.

**Return type** np.ndarray

**easycv.core.post\_processing.pose\_transforms module**

`easycv.core.post_processing.pose_transforms.fliplr_joints(joints_3d, joints_3d_visible, img_width, flip_pairs)`

Flip human joints horizontally.

---

**Note:** num\_keypoints: K

---

**Parameters**

- **joints\_3d** (`np.ndarray([K, 3])`) – Coordinates of keypoints.
- **joints\_3d\_visible** (`np.ndarray([K, 1])`) – Visibility of keypoints.
- **img\_width** (`int`) – Image width.
- **flip\_pairs** (`list[tuple()]`) – Pairs of keypoints which are mirrored (for example, left ear – right ear).

**Returns**

Flipped human joints.

- **joints\_3d\_flipped** (`np.ndarray([K, 3])`): Flipped joints.
- **joints\_3d\_visible\_flipped** (`np.ndarray([K, 1])`): Joint visibility.

**Return type** tuple

`easycv.core.post_processing.pose_transforms.fliplr_regression(regression, flip_pairs, center_mode='static', center_x=0.5, center_index=0)`

Flip human joints horizontally.

---

**Note:** batch\_size: N num\_keypoint: K

---

**Parameters**

- **regression** (`np.ndarray([..., K, C])`) – Coordinates of keypoints, where K is the joint number and C is the dimension. Example shapes are: - [N, K, C]: a batch of keypoints where N is the batch size. - [N, T, K, C]: a batch of pose sequences, where T is the frame number.
- **flip\_pairs** (`list[tuple()]`) – Pairs of keypoints which are mirrored (for example, left ear – right ear).
- **center\_mode** (`str`) – The mode to set the center location on the x-axis to flip around. Options are: - static: use a static x value (see center\_x also) - root: use a root joint (see center\_index also)
- **center\_x** (`float`) – Set the x-axis location of the flip center. Only used when center\_mode=static.
- **center\_index** (`int`) – Set the index of the root joint, whose x location will be used as the flip center. Only used when center\_mode=root.

**Returns**

Flipped human joints.



- `regression_flipped` (`np.ndarray` [..., K, C]): Flipped joints.

**Return type** tuple

`easycv.core.post_processing.pose_transforms.flip_back(output_flipped, flip_pairs, target_type='GaussianHeatmap')`

Flip the flipped heatmaps back to the original form.

---

**Note:** batch\_size: N num\_keypoints: K heatmap height: H heatmap width: W

---

#### Parameters

- **output\_flipped** (`np.ndarray`[N, K, H, W]) – The output heatmaps obtained from the flipped images.
- **flip\_pairs** (`list[tuple()]`) – Pairs of keypoints which are mirrored (for example, left ear – right ear).
- **target\_type** (`str`) – GaussianHeatmap or CombinedTarget

**Returns** heatmaps that flipped back to the original image

**Return type** `np.ndarray`

`easycv.core.post_processing.pose_transforms.transform_preds(coords, center, scale, output_size, use_udp=False)`

Get final keypoint predictions from heatmaps and apply scaling and translation to map them back to the image.

---

**Note:** num\_keypoints: K

---

#### Parameters

- **coords** (`np.ndarray`[K, ndims]) –
  - If ndims=2, corrd are predicted keypoint location.
  - If ndims=4, corrd are composed of (x, y, scores, tags)
  - If ndims=5, corrd are composed of (x, y, scores, tags, flipped\_tags)
- **center** (`np.ndarray`[2, ]) – Center of the bounding box (x, y).
- **scale** (`np.ndarray`[2, ]) – Scale of the bounding box wrt [width, height].
- **output\_size** (`np.ndarray`[2, ] | `list(2,)`) – Size of the destination heatmaps.
- **use\_udp** (`bool`) – Use unbiased data processing

**Returns** Predicted coordinates in the images.

**Return type** `np.ndarray`

`easycv.core.post_processing.pose_transforms.get_affine_transform(center, scale, rot, output_size, shift=(0.0, 0.0), inv=False)`

Get the affine transform matrix, given the center/scale/rot/output\_size.

#### Parameters

- **center** (`np.ndarray`[2, ]) – Center of the bounding box (x, y).
- **scale** (`np.ndarray`[2, ]) – Scale of the bounding box wrt [width, height].
- **rot** (`float`) – Rotation angle (degree).

- **output\_size** (*np.ndarray*[2, ] | *list*(2,)) – Size of the destination heatmaps.
- **shift** (0-100%) – Shift translation ratio wrt the width/height. Default (0., 0.).
- **inv** (*bool*) – Option to inverse the affine transform direction. (inv=False: src->dst or inv=True: dst->src)

**Returns** The transform matrix.

**Return type** *np.ndarray*

`easycv.core.post_processing.pose_transforms.affine_transform(pt, trans_mat)`

Apply an affine transformation to the points.

**Parameters**

- **pt** (*np.ndarray*) – a 2 dimensional point to be transformed
- **trans\_mat** (*np.ndarray*) – 2x3 matrix of an affine transform

**Returns** Transformed points.

**Return type** *np.ndarray*

`easycv.core.post_processing.pose_transforms.rotate_point(pt, angle_rad)`

Rotate a point by an angle.

**Parameters**

- **pt** (*list*[*float*]) – 2 dimensional point to be rotated
- **angle\_rad** (*float*) – rotation angle by radian

**Returns** Rotated point.

**Return type** *list*[*float*]

`easycv.core.post_processing.pose_transforms.get_warp_matrix(theta, size_input, size_dst, size_target)`

Calculate the transformation matrix under the constraint of unbiased. Paper ref: Huang et al. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation (CVPR 2020).

**Parameters**

- **theta** (*float*) – Rotation angle in degrees.
- **size\_input** (*np.ndarray*) – Size of input image [w, h].
- **size\_dst** (*np.ndarray*) – Size of output image [w, h].
- **size\_target** (*np.ndarray*) – Size of ROI in input plane [w, h].

**Returns** A matrix for transformation.

**Return type** *matrix* (*np.ndarray*)

`easycv.core.post_processing.pose_transforms.warp_affine_joints(joints, mat)`

Apply affine transformation defined by the transform matrix on the joints.

**Parameters**

- **joints** (*np.ndarray*[... , 2]) – Origin coordinate of joints.
- **mat** (*np.ndarray*[3, 2]) – The affine matrix.

**Returns** Result coordinate of joints.

**Return type** *matrix* (*np.ndarray*[... , 2])

### 15.1.4 easycv.core.visualization package

`easycv.core.visualization.imshow_bboxes`(*img, bboxes, labels=None, colors='green', text\_color='white', font\_size=20, thickness=1, font\_scale=0.5, show=True, win\_name='', wait\_time=0, out\_file=None*)

Draw bboxes with labels (optional) on an image. This is a wrapper of `mmcv.imshow_bboxes`.

#### Parameters

- **img** (*str or ndarray*) – The image to be displayed.
- **bboxes** (*ndarray*) – `ndarray` of shape (k, 4), each row is a bbox in format [x1, y1, x2, y2].
- **labels** (*str or list[str], optional*) – labels of each bbox.
- **colors** (*list[str or tuple or Color]*) – A list of colors.
- **text\_color** (*str or tuple or Color*) – Color of texts.
- **font\_size** (*int*) – Size of font.
- **thickness** (*int*) – Thickness of lines.
- **font\_scale** (*float*) – Font scales of texts.
- **show** (*bool*) – Whether to show the image.
- **win\_name** (*str*) – The window name.
- **wait\_time** (*int*) – Value of `waitKey` param.
- **out\_file** (*str, optional*) – The filename to write the image.

**Returns** The image with bboxes drawn on it.

**Return type** `ndarray`

`easycv.core.visualization.imshow_keypoints`(*img, pose\_result, skeleton=None, kpt\_score\_thr=0.3, pose\_kpt\_color=None, pose\_link\_color=None, radius=4, thickness=1, show\_keypoint\_weight=False*)

Draw keypoints and links on an image.

#### Parameters

- **img** (*str or Tensor*) – The image to draw poses on. If an image array is given, it will be modified in-place.
- **pose\_result** (*list[kpts]*) – The poses to draw. Each element `kpts` is a set of K keypoints as an `Kx3 numpy.ndarray`, where each keypoint is represented as x, y, score.
- **kpt\_score\_thr** (*float, optional*) – Minimum score of keypoints to be shown. Default: 0.3.
- **pose\_kpt\_color** (*np.array[Nx3]*) – Color of N keypoints. If `None`, the keypoint will not be drawn.
- **pose\_link\_color** (*np.array[Mx3]*) – Color of M links. If `None`, the links will not be drawn.
- **thickness** (*int*) – Thickness of lines.

`easycv.core.visualization.imshow_label`(*img, labels, text\_color='blue', font\_size=20, thickness=1, font\_scale=0.5, interval=5, show=True, win\_name='', wait\_time=0, out\_file=None*)

Draw images with labels on an image.

#### Parameters

- **img** (*str* or *ndarray*) – The image to be displayed.
- **labels** (*str* or *list[str]*) – labels of each image.
- **text\_color** (*str* or *tuple* or *Color*) – Color of texts.
- **font\_size** (*int*) – Size of font.
- **thickness** (*int*) – Thickness of lines.
- **font\_scale** (*float*) – Font scales of texts.
- **intervalint**) – interval pixels between multiple labels
- **show** (*bool*) – Whether to show the image.
- **win\_name** (*str*) – The window name.
- **wait\_time** (*int*) – Value of waitKey param.
- **out\_file** (*str*, *optional*) – The filename to write the image.

**Returns** The image with bboxes drawn on it.

**Return type** *ndarray*

## Submodules

### **easycv.core.visualization.image module**

`easycv.core.visualization.image.get_font_path()`

`easycv.core.visualization.image.put_text(img, xy, text, fill, size=20)`  
support chinese text

`easycv.core.visualization.image.imshow_label(img, labels, text_color='blue', font_size=20, thickness=1,  
font_scale=0.5, interval=5, show=True, win_name="",  
wait_time=0, out_file=None)`

Draw images with labels on an image.

#### **Parameters**

- **img** (*str* or *ndarray*) – The image to be displayed.
- **labels** (*str* or *list[str]*) – labels of each image.
- **text\_color** (*str* or *tuple* or *Color*) – Color of texts.
- **font\_size** (*int*) – Size of font.
- **thickness** (*int*) – Thickness of lines.
- **font\_scale** (*float*) – Font scales of texts.
- **intervalint**) – interval pixels between multiple labels
- **show** (*bool*) – Whether to show the image.
- **win\_name** (*str*) – The window name.
- **wait\_time** (*int*) – Value of waitKey param.
- **out\_file** (*str*, *optional*) – The filename to write the image.

**Returns** The image with bboxes drawn on it.

**Return type** *ndarray*

```
easycv.core.visualization.image.imshow_bboxes(img, bboxes, labels=None, colors='green',
 text_color='white', font_size=20, thickness=1,
 font_scale=0.5, show=True, win_name="", wait_time=0,
 out_file=None)
```

Draw bboxes with labels (optional) on an image. This is a wrapper of `mmcv.imshow_bboxes`.

#### Parameters

- **img** (*str* or *ndarray*) – The image to be displayed.
- **bboxes** (*ndarray*) – *ndarray* of shape (k, 4), each row is a bbox in format [x1, y1, x2, y2].
- **labels** (*str* or *list[str]*, *optional*) – labels of each bbox.
- **colors** (*list[str or tuple or Color]*) – A list of colors.
- **text\_color** (*str* or *tuple* or *Color*) – Color of texts.
- **font\_size** (*int*) – Size of font.
- **thickness** (*int*) – Thickness of lines.
- **font\_scale** (*float*) – Font scales of texts.
- **show** (*bool*) – Whether to show the image.
- **win\_name** (*str*) – The window name.
- **wait\_time** (*int*) – Value of `waitKey` param.
- **out\_file** (*str*, *optional*) – The filename to write the image.

**Returns** The image with bboxes drawn on it.

**Return type** *ndarray*

```
easycv.core.visualization.image.imshow_keypoints(img, pose_result, skeleton=None, kpt_score_thr=0.3,
 pose_kpt_color=None, pose_link_color=None,
 radius=4, thickness=1,
 show_keypoint_weight=False)
```

Draw keypoints and links on an image.

#### Parameters

- **img** (*str* or *Tensor*) – The image to draw poses on. If an image array is given, it will be modified in-place.
- **pose\_result** (*list[kpts]*) – The poses to draw. Each element *kpts* is a set of K keypoints as an *Kx3* *numpy.ndarray*, where each keypoint is represented as x, y, score.
- **kpt\_score\_thr** (*float*, *optional*) – Minimum score of keypoints to be shown. Default: 0.3.
- **pose\_kpt\_color** (*np.array[Nx3]*) – Color of N keypoints. If None, the keypoint will not be drawn.
- **pose\_link\_color** (*np.array[Mx3]*) – Color of M links. If None, the links will not be drawn.
- **thickness** (*int*) – Thickness of lines.

## 15.2 Submodules

### 15.3 easycv.core.standard\_fields module

Contains classes specifying naming conventions used for object detection.

**Specifies:** `InputDataFields`: standard fields used by reader/preprocessor/batcher. `DetectionResultFields`: standard fields returned by object detector. `BoxListFields`: standard field used by `BoxList`. `TfExampleFields`: standard fields for tf-example data format (go/tf-example).

**class** `easycv.core.standard_fields.InputDataFields`

Bases: `object`

Names for the input tensors.

Holds the standard data field names to use for identifying input tensors. This should be used by the decoder to identify keys for the returned `tensor_dict` containing input tensors. And it should be used by the model to identify the tensors it needs.

**image**

image.

**original\_image**

image in the original input size.

**key**

unique key corresponding to image.

**source\_id**

source of the original image.

**filename**

original filename of the dataset (without common path).

**groundtruth\_image\_classes**

image-level class labels.

**groundtruth\_boxes**

coordinates of the ground truth boxes in the image.

**groundtruth\_classes**

box-level class labels.

**groundtruth\_label\_types**

box-level label types (e.g. explicit negative).

**groundtruth\_is\_crowd**

[DEPRECATED, use `groundtruth_group_of` instead] is the groundtruth a single object or a crowd.

**groundtruth\_area**

area of a groundtruth segment.

**groundtruth\_difficult**

is a *difficult* object

**groundtruth\_group\_of**

is a *group\_of* objects, e.g. multiple objects of the same class, forming a connected group, where instances are heavily occluding each other.

**proposal\_boxes**

coordinates of object proposal boxes.

**proposal\_objectness**  
objectness score of each proposal.

**groundtruth\_instance\_masks**  
ground truth instance masks.

**groundtruth\_instance\_boundaries**  
ground truth instance boundaries.

**groundtruth\_instance\_classes**  
instance mask-level class labels.

**groundtruth\_keypoints**  
ground truth keypoints.

**groundtruth\_keypoint\_visibilities**  
ground truth keypoint visibilities.

**groundtruth\_label\_scores**  
groundtruth label scores.

**groundtruth\_weights**  
groundtruth weight factor for bounding boxes.

**num\_groundtruth\_boxes**  
number of groundtruth boxes.

**true\_image\_shapes**  
true shapes of images in the resized images, as resized images can be padded with zeros.

**image** = 'image'

**mask** = 'mask'

**width** = 'width'

**height** = 'height'

**original\_image** = 'original\_image'

**optical\_flow** = 'optical\_flow'

**key** = 'key'

**source\_id** = 'source\_id'

**filename** = 'filename'

**dataset\_name** = 'dataset\_name'

**groundtruth\_image\_classes** = 'groundtruth\_image\_classes'

**groundtruth\_image\_classes\_num** = 'groundtruth\_image\_classes\_num'

**groundtruth\_boxes** = 'groundtruth\_boxes'

**groundtruth\_classes** = 'groundtruth\_classes'

**groundtruth\_label\_types** = 'groundtruth\_label\_types'

**groundtruth\_is\_crowd** = 'groundtruth\_is\_crowd'

**groundtruth\_area** = 'groundtruth\_area'

**groundtruth\_difficult** = 'groundtruth\_difficult'

**groundtruth\_group\_of** = 'groundtruth\_group\_of'

```
proposal_boxes = 'proposal_boxes'
proposal_objectness = 'proposal_objectness'
groundtruth_instance_masks = 'groundtruth_instance_masks'
groundtruth_instance_boundaries = 'groundtruth_instance_boundaries'
groundtruth_instance_classes = 'groundtruth_instance_classes'
groundtruth_keypoints = 'groundtruth_keypoints'
groundtruth_keypoint_visibilities = 'groundtruth_keypoint_visibilities'
groundtruth_label_scores = 'groundtruth_label_scores'
groundtruth_weights = 'groundtruth_weights'
num_groundtruth_boxes = 'num_groundtruth_boxes'
true_image_shape = 'true_image_shape'
original_image_shape = 'original_image_shape'
original_instance_masks = 'original_instance_masks'
groundtruth_boxes_absolute = 'groundtruth_boxes_absolute'
groundtruth_keypoints_absolute = 'groundtruth_keypoints_absolute'
label_map = 'label_map'
char_dict = 'char_dict'
```

```
class easycv.core.standard_fields.DetectionResultFields
```

```
 Bases: object
```

```
 Naming conventions for storing the output of the detector.
```

```
 source_id
```

```
 source of the original image.
```

```
 key
```

```
 unique key corresponding to image.
```

```
 detection_boxes
```

```
 coordinates of the detection boxes in the image.
```

```
 detection_scores
```

```
 detection scores for the detection boxes in the image.
```

```
 detection_classes
```

```
 detection-level class labels.
```

```
 detection_masks
```

```
 contains a segmentation mask for each detection box.
```

```
 detection_boundaries
```

```
 contains an object boundary for each detection box.
```

```
 detection_keypoints
```

```
 contains detection keypoints for each detection box.
```

```
 num_detections
```

```
 number of detections in the batch.
```

```
 source_id = 'source_id'
```



```

key = 'key'
detection_boxes = 'detection_boxes'
detection_scores = 'detection_scores'
detection_classes = 'detection_classes'
detection_masks = 'detection_masks'
detection_boundaries = 'detection_boundaries'
detection_keypoints = 'detection_keypoints'
num_detections = 'num_detections'

```

**class** easycv.core.standard\_fields.TfExampleFields

Bases: object

TF-example proto feature names for object detection.

Holds the standard feature names to load from an Example proto for object detection.

**image\_encoded**  
JPEG encoded string

**image\_format**  
image format, e.g. “JPEG”

**filename**  
filename

**channels**  
number of channels of image

**colorspace**  
colorspace, e.g. “RGB”

**height**  
height of image in pixels, e.g. 462

**width**  
width of image in pixels, e.g. 581

**source\_id**  
original source of the image

**object\_class\_text**  
labels in text format, e.g. [“person”, “cat”]

**object\_class\_label**  
labels in numbers, e.g. [16, 8]

**object\_bbox\_xmin**  
xmin coordinates of groundtruth box, e.g. 10, 30

**object\_bbox\_xmax**  
xmax coordinates of groundtruth box, e.g. 50, 40

**object\_bbox\_ymin**  
ymin coordinates of groundtruth box, e.g. 40, 50

**object\_bbox\_ymax**  
ymax coordinates of groundtruth box, e.g. 80, 70

**object\_view**  
viewpoint of object, e.g. ["frontal", "left"]

**object\_truncated**  
is object truncated, e.g. [true, false]

**object\_occluded**  
is object occluded, e.g. [true, false]

**object\_difficult**  
is object difficult, e.g. [true, false]

**object\_group\_of**  
is object a single object or a group of objects

**object\_depiction**  
is object a depiction

**object\_is\_crowd**  
[DEPRECATED, use object\_group\_of instead] is the object a single object or a crowd

**object\_segment\_area**  
the area of the segment.

**object\_weight**  
a weight factor for the object's bounding box.

**instance\_masks**  
instance segmentation masks.

**instance\_boundaries**  
instance boundaries.

**instance\_classes**  
Classes for each instance segmentation mask.

**detection\_class\_label**  
class label in numbers.

**detection\_bbox\_ymin**  
ymin coordinates of a detection box.

**detection\_bbox\_xmin**  
xmin coordinates of a detection box.

**detection\_bbox\_ymax**  
ymax coordinates of a detection box.

**detection\_bbox\_xmax**  
xmax coordinates of a detection box.

**detection\_score**  
detection score for the class label and box.

**image\_encoded** = 'image/encoded'

**image\_format** = 'image/format'

**filename** = 'image/filename'

**channels** = 'image/channels'

**colorspace** = 'image/colorspace'

**height** = 'image/height'

```
width = 'image/width'
source_id = 'image/source_id'
object_class_text = 'image/object/class/text'
object_class_label = 'image/object/class/label'
object_bbox_ymin = 'image/object/bbox/ymin'
object_bbox_xmin = 'image/object/bbox/xmin'
object_bbox_ymax = 'image/object/bbox/ymax'
object_bbox_xmax = 'image/object/bbox/xmax'
object_view = 'image/object/view'
object_truncated = 'image/object/truncated'
object_occluded = 'image/object/occluded'
object_difficult = 'image/object/difficult'
object_group_of = 'image/object/group_of'
object_depiction = 'image/object/depiction'
object_is_crowd = 'image/object/is_crowd'
object_segment_area = 'image/object/segment/area'
object_weight = 'image/object/weight'
instance_masks = 'image/segmentation/object'
instance_boundaries = 'image/boundaries/object'
instance_classes = 'image/segmentation/object/class'
detection_class_label = 'image/detection/label'
detection_bbox_ymin = 'image/detection/bbox/ymin'
detection_bbox_xmin = 'image/detection/bbox/xmin'
detection_bbox_ymax = 'image/detection/bbox/ymax'
detection_bbox_xmax = 'image/detection/bbox/xmax'
detection_score = 'image/detection/score'
```



## EASYCV.MODELS PACKAGE

### 16.1 Subpackages

#### 16.1.1 easycv.models.backbones package

##### Submodules

##### easycv.models.backbones.benchmark\_mlp module

```
class easycv.models.backbones.benchmark_mlp.BenchMarkMLP(feature_num, num_classes=1000,
 avg_pool=False, **kwargs)
```

Bases: torch.nn.modules.module.Module

```
__init__(feature_num, num_classes=1000, avg_pool=False, **kwargs)
 Initializes internal Module state, shared by both nn.Module and ScriptModule.
```

```
init_weights(pretrained=None)
```

```
forward(x)
 Defines the computation performed at every call.
 Should be overridden by all subclasses.
```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
training: bool
```

##### easycv.models.backbones.bninception module

This model is taken from the official PyTorch model zoo. - torchvision.models.mobilenet.py on 31th Aug, 2019

```
class easycv.models.backbones.bninception.BNInception(num_classes=0)
```

Bases: torch.nn.modules.module.Module

```
__init__(num_classes=0)
 Initializes internal Module state, shared by both nn.Module and ScriptModule.
```

```
init_weights(pretrained=None)
```

```
features(input)
```

**logits**(*features*)

**forward**(*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

### **easycv.models.backbones.darknet module**

**class** easycv.models.backbones.darknet.**Darknet**(*depth*, *in\_channels*=3, *stem\_out\_channels*=32, *out\_features*=('dark3', 'dark4', 'dark5'))

Bases: torch.nn.modules.module.Module

**depth2blocks** = {21: [1, 2, 2, 1], 53: [2, 8, 8, 4]}

**\_\_init\_\_**(*depth*, *in\_channels*=3, *stem\_out\_channels*=32, *out\_features*=('dark3', 'dark4', 'dark5'))

#### **Parameters**

- **depth** (*int*) – depth of darknet used in model, usually use [21, 53] for this param.
- **in\_channels** (*int*) – number of input channels, for example, use 3 for RGB image.
- **stem\_out\_channels** (*int*) – number of output channels of darknet stem. It decides channels of darknet layer2 to layer5.
- **out\_features** (*Tuple[str]*) – desired output layer name.

**make\_group\_layer**(*in\_channels*: *int*, *num\_blocks*: *int*, *stride*: *int* = 1)

starts with conv layer then has *num\_blocks* ResLayer

**make\_spp\_block**(*filters\_list*, *in\_filters*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** easycv.models.backbones.darknet.**CSPDarknet**(*dep\_mul*, *wid\_mul*, *out\_features*=('dark3', 'dark4', 'dark5'), *depthwise*=False, *act*='silu')

Bases: torch.nn.modules.module.Module

**\_\_init\_\_**(*dep\_mul*, *wid\_mul*, *out\_features*=('dark3', 'dark4', 'dark5'), *depthwise*=False, *act*='silu')

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**easycv.models.backbones.genet module**

easycv.models.backbones.genet.remove\_bn\_in\_superblock(*super\_block*)

easycv.models.backbones.genet.fuse\_bn(*model*)

**class** easycv.models.backbones.genet.PlainNetBasicBlockClass(*in\_channels=0, out\_channels=0, stride=1, no\_create=False, block\_name=None, \*\*kwargs*)

Bases: torch.nn.modules.module.Module

**\_\_init\_\_**(*in\_channels=0, out\_channels=0, stride=1, no\_create=False, block\_name=None, \*\*kwargs*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**static create\_from\_str**(*s, no\_create=False*)

**static is\_instance\_from\_str**(*s*)

**training:** bool

**class** easycv.models.backbones.genet.AdaptiveAvgPool(*out\_channels, output\_size, no\_create=False, block\_name=None, \*\*kwargs*)

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

**\_\_init\_\_**(*out\_channels, output\_size, no\_create=False, block\_name=None, \*\*kwargs*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
static create_from_str(s, no_create=False)
static is_instance_from_str(s)
training: bool
class easycv.models.backbones.genet.BN(out_channels=None, copy_from=None, no_create=False,
 block_name=None, **kwargs)
 Bases: easycv.models.backbones.genet.PlainNetBasicBlockClass
 __init__(out_channels=None, copy_from=None, no_create=False, block_name=None, **kwargs)
 Initializes internal Module state, shared by both nn.Module and ScriptModule.
 forward(x)
 Defines the computation performed at every call.
 Should be overridden by all subclasses.
```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
static create_from_str(s, no_create=False)
static is_instance_from_str(s)
training: bool
class easycv.models.backbones.genet.ConvDW(out_channels=None, kernel_size=None, stride=None,
 copy_from=None, no_create=False, block_name=None,
 **kwargs)
 Bases: easycv.models.backbones.genet.PlainNetBasicBlockClass
 __init__(out_channels=None, kernel_size=None, stride=None, copy_from=None, no_create=False,
 block_name=None, **kwargs)
 Initializes internal Module state, shared by both nn.Module and ScriptModule.
 forward(x)
 Defines the computation performed at every call.
 Should be overridden by all subclasses.
```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
static create_from_str(s, no_create=False)
static is_instance_from_str(s)
training: bool
class easycv.models.backbones.genet.ConvKX(in_channels=None, out_channels=None, kernel_size=None,
 stride=None, copy_from=None, no_create=False,
 block_name=None, **kwargs)
 Bases: easycv.models.backbones.genet.PlainNetBasicBlockClass
 __init__(in_channels=None, out_channels=None, kernel_size=None, stride=None, copy_from=None,
 no_create=False, block_name=None, **kwargs)
 Initializes internal Module state, shared by both nn.Module and ScriptModule.
```



**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**static create\_from\_str**(s, no\_create=False)

**static is\_instance\_from\_str**(s)

**training:** bool

**class** easycv.models.backbones.genet.Flatten(out\_channels, no\_create=False, block\_name=None, \*\*kwargs)

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

**\_\_init\_\_**(out\_channels, no\_create=False, block\_name=None, \*\*kwargs)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**static create\_from\_str**(s, no\_create=False)

**static is\_instance\_from\_str**(s)

**training:** bool

**class** easycv.models.backbones.genet.Linear(in\_channels=None, out\_channels=None, bias=None, copy\_from=None, no\_create=False, block\_name=None, \*\*kwargs)

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

**\_\_init\_\_**(in\_channels=None, out\_channels=None, bias=None, copy\_from=None, no\_create=False, block\_name=None, \*\*kwargs)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**static create\_from\_str**(s, no\_create=False)

**static is\_instance\_from\_str**(s)

**training:** bool

**class** easycv.models.backbones.genet.**MaxPool**(*out\_channels, kernel\_size, stride, no\_create=False, block\_name=None, \*\*kwargs*)

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

**\_\_init\_\_**(*out\_channels, kernel\_size, stride, no\_create=False, block\_name=None, \*\*kwargs*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**static create\_from\_str**(*s, no\_create=False*)

**static is\_instance\_from\_str**(*s*)

**training:** bool

**class** easycv.models.backbones.genet.**MultiSumBlock**(*inner\_block\_list, no\_create=False, block\_name=None, \*\*kwargs*)

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

**\_\_init\_\_**(*inner\_block\_list, no\_create=False, block\_name=None, \*\*kwargs*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**static create\_from\_str**(*s, no\_create=False*)

**static is\_instance\_from\_str**(*s*)

**training:** bool

**class** easycv.models.backbones.genet.**RELU**(*out\_channels, no\_create=False, block\_name=None, \*\*kwargs*)

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

**\_\_init\_\_**(*out\_channels, no\_create=False, block\_name=None, \*\*kwargs*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```

static create_from_str(s, no_create=False)
static is_instance_from_str(s)
training: bool
class easycv.models.backbones.genet.ResBlock(inner_block_list, in_channels=None, stride=None,
 no_create=False, block_name=None, **kwargs)
 Bases: easycv.models.backbones.genet.PlainNetBasicBlockClass
 ResBlock(in_channels, inner_blocks_str). If in_channels is missing, use inner_block_list[0].in_channels as
 in_channels
 __init__(inner_block_list, in_channels=None, stride=None, no_create=False, block_name=None,
 **kwargs)
 Initializes internal Module state, shared by both nn.Module and ScriptModule.
 forward(x)
 Defines the computation performed at every call.
 Should be overridden by all subclasses.

```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```

static create_from_str(s, no_create=False)
static is_instance_from_str(s)
training: bool
class easycv.models.backbones.genet.Sequential(inner_block_list, no_create=False, block_name=None,
 **kwargs)
 Bases: easycv.models.backbones.genet.PlainNetBasicBlockClass
 __init__(inner_block_list, no_create=False, block_name=None, **kwargs)
 Initializes internal Module state, shared by both nn.Module and ScriptModule.
 forward(x)
 Defines the computation performed at every call.
 Should be overridden by all subclasses.

```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```

static create_from_str(s, no_create=False)
static is_instance_from_str(s)
training: bool

```

```
class easycv.models.backbones.genet.SuperResXXXX(in_channels=0, out_channels=0, kernel_size=3,
 stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

```
__init__(in_channels=0, out_channels=0, kernel_size=3, stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
static create_from_str(s, no_create=False)
```

```
static is_instance_from_str(s)
```

```
training: bool
```

```
class easycv.models.backbones.genet.SuperResK1KX(in_channels=0, out_channels=0, kernel_size=3,
 stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

```
__init__(in_channels=0, out_channels=0, kernel_size=3, stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
static create_from_str(s, no_create=False)
```

```
static is_instance_from_str(s)
```

```
training: bool
```

```
class easycv.models.backbones.genet.SuperResK1KXK1(in_channels=0, out_channels=0, kernel_size=3,
 stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

```
__init__(in_channels=0, out_channels=0, kernel_size=3, stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
static create_from_str(s, no_create=False)
```

```
static is_instance_from_str(s)
```

```
training: bool
```

```
class easycv.models.backbones.genet.SuperResK1DWK1(in_channels=0, out_channels=0, kernel_size=3,
 stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

```
__init__(in_channels=0, out_channels=0, kernel_size=3, stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
static create_from_str(s, no_create=False)
```

```
static is_instance_from_str(s)
```

```
training: bool
```

```
class easycv.models.backbones.genet.SuperResK1DW(in_channels=0, out_channels=0, kernel_size=3,
 stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Bases: [easycv.models.backbones.genet.PlainNetBasicBlockClass](#)

```
__init__(in_channels=0, out_channels=0, kernel_size=3, stride=1, expansion=1.0, sublayers=1,
 no_create=False, block_name=None, **kwargs)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
training: bool
```

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
static create_from_str(s, no_create=False)
```

```
static is_instance_from_str(s)

class easycv.models.backbones.genet.PlainNet(plainnet_struct_idx=None, num_classes=0,
 no_create=False, **kwargs)

Bases: torch.nn.modules.module.Module

training: bool

__init__(plainnet_struct_idx=None, num_classes=0, no_create=False, **kwargs)
 Initializes internal Module state, shared by both nn.Module and ScriptModule.

init_weights(pretrained=None)

forward(x)
 Defines the computation performed at every call.

 Should be overridden by all subclasses.
```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

## easycv.models.backbones.hrnet module

```
easycv.models.backbones.hrnet.get_expansion(block, expansion=None)

Get the expansion of a residual block.
```

The block expansion will be obtained by the following order:

1. If `expansion` is given, just return it.
2. If `block` has the attribute `expansion`, then return `block.expansion`.
3. Return the default value according the the block type: 1 for `BasicBlock` and 4 for `Bottleneck`.

### Parameters

- **block** (*class*) – The block class.
- **expansion** (*int* / *None*) – The given expansion ratio.

**Returns** The expansion of the block.

**Return type** `int`

```
class easycv.models.backbones.hrnet.Bottleneck(in_channels, out_channels, expansion=4, stride=1,
 dilation=1, downsample=None, style='pytorch',
 with_cp=False, conv_cfg=None, norm_cfg={'type':
 'BN'})
```

Bases: `torch.nn.modules.module.Module`

Bottleneck block for ResNet.

### Parameters

- **in\_channels** (*int*) – Input channels of this block.
- **out\_channels** (*int*) – Output channels of this block.
- **expansion** (*int*) – The ratio of `out_channels/mid_channels` where `mid_channels` is the input/output channels of conv2. Default: 4.
- **stride** (*int*) – stride of the block. Default: 1
- **dilation** (*int*) – dilation of convolution. Default: 1

- **downsample** (*nn.Module*) – downsample operation on identity branch. Default: None.
- **style** (*str*) – "pytorch" or "caffe". If set to "pytorch", the stride-two layer is the 3x3 conv layer, otherwise the stride-two layer is the first 1x1 conv layer. Default: "pytorch".
- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed.
- **conv\_cfg** (*dict*) – dictionary to construct and config conv layer. Default: None
- **norm\_cfg** (*dict*) – dictionary to construct and config norm layer. Default: dict(type='BN')

**\_\_init\_\_** (*in\_channels, out\_channels, expansion=4, stride=1, dilation=1, downsample=None, style='pytorch', with\_cp=False, conv\_cfg=None, norm\_cfg={'type': 'BN'}*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**property norm1**

the normalization layer named "norm1"

**Type** nn.Module

**property norm2**

the normalization layer named "norm2"

**Type** nn.Module

**property norm3**

the normalization layer named "norm3"

**Type** nn.Module

**forward** (*x*)

Forward function.

**training:** bool

**class** easycv.models.backbones.hrnet.**HRModule** (*num\_branches, blocks, num\_blocks, in\_channels, num\_channels, multiscale\_output=False, with\_cp=False, conv\_cfg=None, norm\_cfg={'type': 'BN'}, upsample\_cfg={'align\_corners': None, 'mode': 'nearest'}*)

Bases: torch.nn.modules.module.Module

High-Resolution Module for HRNet.

In this module, every branch has 4 BasicBlocks/Bottlenecks. Fusion/Exchange is in this module.

**\_\_init\_\_** (*num\_branches, blocks, num\_blocks, in\_channels, num\_channels, multiscale\_output=False, with\_cp=False, conv\_cfg=None, norm\_cfg={'type': 'BN'}, upsample\_cfg={'align\_corners': None, 'mode': 'nearest'}*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*)

Forward function.

**training:** bool

**class** easycv.models.backbones.hrnet.**HRNet** (*arch='w32', extra=None, in\_channels=3, conv\_cfg=None, norm\_cfg={'type': 'BN'}, norm\_eval=False, with\_cp=False, zero\_init\_residual=False, multi\_scale\_output=False)*

Bases: torch.nn.modules.module.Module

HRNet backbone.

## High-Resolution Representations for Labeling Pixels and Regions

## Parameters

- **extra** (*dict*) – detailed configuration for each stage of HRNet.
- **in\_channels** (*int*) – Number of input image channels. Default: 3.
- **conv\_cfg** (*dict*) – dictionary to construct and config conv layer.
- **norm\_cfg** (*dict*) – dictionary to construct and config norm layer.
- **norm\_eval** (*bool*) – Whether to set norm layers to eval mode, namely, freeze running stats (mean and var). Note: Effect on Batch Norm and its variants only. Default: False
- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed.
- **zero\_init\_residual** (*bool*) – whether to use zero init for last norm layer in resblocks to let them behave as identity.

## Example

```
>>> from mmpose.models import HRNet
>>> import torch
>>> extra = dict(
>>> stage1=dict(
>>> num_modules=1,
>>> num_branches=1,
>>> block='BOTTLENECK',
>>> num_blocks=(4,),
>>> num_channels=(64,)),
>>> stage2=dict(
>>> num_modules=1,
>>> num_branches=2,
>>> block='BASIC',
>>> num_blocks=(4, 4),
>>> num_channels=(32, 64)),
>>> stage3=dict(
>>> num_modules=4,
>>> num_branches=3,
>>> block='BASIC',
>>> num_blocks=(4, 4, 4),
>>> num_channels=(32, 64, 128)),
>>> stage4=dict(
>>> num_modules=3,
>>> num_branches=4,
>>> block='BASIC',
>>> num_blocks=(4, 4, 4, 4),
>>> num_channels=(32, 64, 128, 256)))
>>> self = HRNet(extra, in_channels=1)
>>> self.eval()
>>> inputs = torch.rand(1, 1, 32, 32)
>>> level_outputs = self.forward(inputs)
>>> for level_out in level_outputs:
... print(tuple(level_out.shape))
(1, 32, 8, 8)
```



```

blocks_dict = {'BASIC': <class 'easycv.models.backbones.resnet.BasicBlock'>,
'BOTTLENECK': <class 'easycv.models.backbones.hrnet.Bottleneck'>}

arch_zoo = {'w18': [[1, 1, 'BOTTLENECK', (4,), (64,)], [1, 2, 'BASIC', (4, 4), (18,
36)], [4, 3, 'BASIC', (4, 4, 4), (18, 36, 72)], [3, 4, 'BASIC', (4, 4, 4, 4), (18,
36, 72, 144)]], 'w30': [[1, 1, 'BOTTLENECK', (4,), (64,)], [1, 2, 'BASIC', (4, 4),
(30, 60)], [4, 3, 'BASIC', (4, 4, 4), (30, 60, 120)], [3, 4, 'BASIC', (4, 4, 4, 4),
(30, 60, 120, 240)]], 'w32': [[1, 1, 'BOTTLENECK', (4,), (64,)], [1, 2, 'BASIC',
(4, 4), (32, 64)], [4, 3, 'BASIC', (4, 4, 4), (32, 64, 128)], [3, 4, 'BASIC', (4, 4,
4, 4), (32, 64, 128, 256)]], 'w40': [[1, 1, 'BOTTLENECK', (4,), (64,)], [1, 2,
'BASIC', (4, 4), (40, 80)], [4, 3, 'BASIC', (4, 4, 4), (40, 80, 160)], [3, 4,
'BASIC', (4, 4, 4, 4), (40, 80, 160, 320)]], 'w44': [[1, 1, 'BOTTLENECK', (4,),
(64,)], [1, 2, 'BASIC', (4, 4), (44, 88)], [4, 3, 'BASIC', (4, 4, 4), (44, 88,
176)], [3, 4, 'BASIC', (4, 4, 4, 4), (44, 88, 176, 352)]], 'w48': [[1, 1,
'BOTTLENECK', (4,), (64,)], [1, 2, 'BASIC', (4, 4), (48, 96)], [4, 3, 'BASIC', (4,
4, 4), (48, 96, 192)], [3, 4, 'BASIC', (4, 4, 4, 4), (48, 96, 192, 384)]], 'w64':
[[1, 1, 'BOTTLENECK', (4,), (64,)], [1, 2, 'BASIC', (4, 4), (64, 128)], [4, 3,
'BASIC', (4, 4, 4), (64, 128, 256)], [3, 4, 'BASIC', (4, 4, 4, 4), (64, 128, 256,
512)]]}

__init__(arch='w32', extra=None, in_channels=3, conv_cfg=None, norm_cfg={'type': 'BN'},
norm_eval=False, with_cp=False, zero_init_residual=False, multi_scale_output=False)
 Initializes internal Module state, shared by both nn.Module and ScriptModule.

property norm1
 the normalization layer named “norm1”

 Type nn.Module

property norm2
 the normalization layer named “norm2”

 Type nn.Module

init_weights(pretrained=None)
 Initialize the weights in backbone.

 Parameters pretrained (str, optional) – Path to pre-trained weights. Defaults to
 None.

forward(x)
 Forward function.

train(mode=True)
 Convert the model into training mode.

training: bool

parse_arch(arch, extra=None)

```

### easycv.models.backbones.inceptionv3 module

This model is taken from the official PyTorch model zoo. - torchvision.models.inception.py on 31th Aug, 2019

```
class easycv.models.backbones.inceptionv3.Inception3(num_classes: int = 0, aux_logits: bool = True,
 transform_input: bool = False)
```

Bases: torch.nn.modules.module.Module

```
__init__(num_classes: int = 0, aux_logits: bool = True, transform_input: bool = False) → None
```

#### Parameters

- **num\_classes** – number of classes based on dataset.
- **aux\_logits** – If True, adds two auxiliary branches that can improve training. Default: *False* when pretrained is True otherwise *True*
- **transform\_input** – If True, preprocesses the input according to the method with which it was trained on ImageNet. Default: *False*

```
init_weights(pretrained=None)
```

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

### easycv.models.backbones.lighthrnet module

```
easycv.models.backbones.lighthrnet.channel_shuffle(x, groups)
```

Channel Shuffle operation.

This function enables cross-group information flow for multiple groups convolution layers.

#### Parameters

- **x** (*Tensor*) – The input tensor.
- **groups** (*int*) – The number of groups to divide the input tensor in the channel dimension.

**Returns** The output tensor after channel shuffle operation.

**Return type** Tensor

```
class easycv.models.backbones.lighthrnet.SpatialWeighting(channels, ratio=16, conv_cfg=None,
 norm_cfg=None, act_cfg=({'type':
 'ReLU'}, {'type': 'Sigmoid'}))
```

Bases: torch.nn.modules.module.Module

Spatial weighting module.

#### Parameters

- **channels** (*int*) – The channels of the module.

- **ratio** (*int*) – channel reduction ratio.
- **conv\_cfg** (*dict*) – Config dict for convolution layer. Default: None, which means using conv2d.
- **norm\_cfg** (*dict*) – Config dict for normalization layer. Default: None.
- **act\_cfg** (*dict*) – Config dict for activation layer. Default: (dict(type='ReLU'), dict(type='Sigmoid')). The last ConvModule uses Sigmoid by default.

**\_\_init\_\_** (*channels, ratio=16, conv\_cfg=None, norm\_cfg=None, act\_cfg=({'type': 'ReLU'}, {'type': 'Sigmoid'})*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.lighthrnet.CrossResolutionWeighting(channels, ratio=16,
 conv_cfg=None,
 norm_cfg=None,
 act_cfg=({'type': 'ReLU'},
 {'type': 'Sigmoid'}))
```

Bases: torch.nn.modules.module.Module

Cross-resolution channel weighting module.

**Parameters**

- **channels** (*int*) – The channels of the module.
- **ratio** (*int*) – channel reduction ratio.
- **conv\_cfg** (*dict*) – Config dict for convolution layer. Default: None, which means using conv2d.
- **norm\_cfg** (*dict*) – Config dict for normalization layer. Default: None.
- **act\_cfg** (*dict*) – Config dict for activation layer. Default: (dict(type='ReLU'), dict(type='Sigmoid')). The last ConvModule uses Sigmoid by default.

**\_\_init\_\_** (*channels, ratio=16, conv\_cfg=None, norm\_cfg=None, act\_cfg=({'type': 'ReLU'}, {'type': 'Sigmoid'})*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.lighthrnet.ConditionalChannelWeighting(in_channels, stride,
 reduce_ratio,
 conv_cfg=None,
 norm_cfg={'type': 'BN'},
 with_cp=False)
```

Bases: `torch.nn.modules.module.Module`

Conditional channel weighting block.

**Parameters**

- **in\_channels** (*int*) – The input channels of the block.
- **stride** (*int*) – Stride of the 3x3 convolution layer.
- **reduce\_ratio** (*int*) – channel reduction ratio.
- **conv\_cfg** (*dict*) – Config dict for convolution layer. Default: None, which means using conv2d.
- **norm\_cfg** (*dict*) – Config dict for normalization layer. Default: dict(type='BN').
- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed. Default: False.

**\_\_init\_\_** (*in\_channels, stride, reduce\_ratio, conv\_cfg=None, norm\_cfg={'type': 'BN'}, with\_cp=False*)  
Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

```
class easycv.models.backbones.lighthrnet.Stem(in_channels, stem_channels, out_channels,
 expand_ratio, conv_cfg=None, norm_cfg={'type': 'BN'},
 with_cp=False)
```

Bases: `torch.nn.modules.module.Module`

Stem network block.

**Parameters**

- **in\_channels** (*int*) – The input channels of the block.
- **stem\_channels** (*int*) – Output channels of the stem layer.
- **out\_channels** (*int*) – The output channels of the block.
- **expand\_ratio** (*int*) – adjusts number of channels of the hidden layer in InvertedResidual by this amount.
- **conv\_cfg** (*dict*) – Config dict for convolution layer. Default: None, which means using conv2d.
- **norm\_cfg** (*dict*) – Config dict for normalization layer. Default: dict(type='BN').
- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed. Default: False.

---

```
__init__(in_channels, stem_channels, out_channels, expand_ratio, conv_cfg=None, norm_cfg={'type': 'BN'}, with_cp=False)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.lighthrnet.IterativeHead(in_channels, norm_cfg={'type': 'BN'})
```

Bases: torch.nn.modules.module.Module

Extra iterative head for feature learning.

**Parameters**

- **in\_channels** (*int*) – The input channels of the block.
- **norm\_cfg** (*dict*) – Config dict for normalization layer. Default: dict(type='BN').

```
__init__(in_channels, norm_cfg={'type': 'BN'})
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.lighthrnet.ShuffleUnit(in_channels, out_channels, stride=1, conv_cfg=None, norm_cfg={'type': 'BN'}, act_cfg={'type': 'ReLU'}, with_cp=False)
```

Bases: torch.nn.modules.module.Module

InvertedResidual block for ShuffleNetV2 backbone.

**Parameters**

- **in\_channels** (*int*) – The input channels of the block.
- **out\_channels** (*int*) – The output channels of the block.
- **stride** (*int*) – Stride of the 3x3 convolution layer. Default: 1
- **conv\_cfg** (*dict*) – Config dict for convolution layer. Default: None, which means using conv2d.
- **norm\_cfg** (*dict*) – Config dict for normalization layer. Default: dict(type='BN').
- **act\_cfg** (*dict*) – Config dict for activation layer. Default: dict(type='ReLU').

- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed. Default: False.

**\_\_init\_\_** (*in\_channels, out\_channels, stride=1, conv\_cfg=None, norm\_cfg={'type': 'BN'}, act\_cfg={'type': 'ReLU'}, with\_cp=False*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** *bool*

**class** easycv.models.backbones.lighthrnet.LiteHRModule (*num\_branches, num\_blocks, in\_channels, reduce\_ratio, module\_type, multiscale\_output=False, with\_fuse=True, conv\_cfg=None, norm\_cfg={'type': 'BN'}, with\_cp=False*)

Bases: torch.nn.modules.module.Module

High-Resolution Module for LiteHRNet.

It contains conditional channel weighting blocks and shuffle blocks.

**Parameters**

- **num\_branches** (*int*) – Number of branches in the module.
- **num\_blocks** (*int*) – Number of blocks in the module.
- **in\_channels** (*list(int)*) – Number of input image channels.
- **reduce\_ratio** (*int*) – Channel reduction ratio.
- **module\_type** (*str*) – ‘LITE’ or ‘NAIVE’
- **multiscale\_output** (*bool*) – Whether to output multi-scale features.
- **with\_fuse** (*bool*) – Whether to use fuse layers.
- **conv\_cfg** (*dict*) – dictionary to construct and config conv layer.
- **norm\_cfg** (*dict*) – dictionary to construct and config norm layer.
- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed.

**\_\_init\_\_** (*num\_branches, num\_blocks, in\_channels, reduce\_ratio, module\_type, multiscale\_output=False, with\_fuse=True, conv\_cfg=None, norm\_cfg={'type': 'BN'}, with\_cp=False*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*)

Forward function.

**training:** *bool*

**class** easycv.models.backbones.lighthrnet.LiteHRNet (*extra, in\_channels=3, conv\_cfg=None, norm\_cfg={'type': 'BN'}, norm\_eval=False, with\_cp=False*)

Bases: torch.nn.modules.module.Module

Lite-HRNet backbone.

Lite-HRNet: A Lightweight High-Resolution Network

Code adapted from '<https://github.com/HRNet/Lite-HRNet/>' 'blob/hrnet/models/backbones/litehrnet.py'

#### Parameters

- **extra** (*dict*) – detailed configuration for each stage of HRNet.
- **in\_channels** (*int*) – Number of input image channels. Default: 3.
- **conv\_cfg** (*dict*) – dictionary to construct and config conv layer.
- **norm\_cfg** (*dict*) – dictionary to construct and config norm layer.
- **norm\_eval** (*bool*) – Whether to set norm layers to eval mode, namely, freeze running stats (mean and var). Note: Effect on Batch Norm and its variants only. Default: False
- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed.

#### Example

```
>>> from mmpose.models import LiteHRNet
>>> import torch
>>> extra=dict(
>>> stem=dict(stem_channels=32, out_channels=32, expand_ratio=1),
>>> num_stages=3,
>>> stages_spec=dict(
>>> num_modules=(2, 4, 2),
>>> num_branches=(2, 3, 4),
>>> num_blocks=(2, 2, 2),
>>> module_type=('LITE', 'LITE', 'LITE'),
>>> with_fuse=(True, True, True),
>>> reduce_ratios=(8, 8, 8),
>>> num_channels=(
>>> (40, 80),
>>> (40, 80, 160),
>>> (40, 80, 160, 320),
>>>)),
>>> with_head=False)
>>> self = LiteHRNet(extra, in_channels=1)
>>> self.eval()
>>> inputs = torch.rand(1, 1, 32, 32)
>>> level_outputs = self.forward(inputs)
>>> for level_out in level_outputs:
... print(tuple(level_out.shape))
(1, 40, 8, 8)
```

**\_\_init\_\_** (*extra*, *in\_channels*=3, *conv\_cfg*=None, *norm\_cfg*={'type': 'BN'}, *norm\_eval*=False, *with\_cp*=False)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights** (*pretrained*=None)

Initialize the weights in backbone.

**Parameters pretrained** (*str*, *optional*) – Path to pre-trained weights. Defaults to None.

**forward**(*x*)  
Forward function.

**train**(*mode=True*)  
Convert the model into training mode.

**training:** `bool`

### **easycv.models.backbones.mae\_vit\_transformer module**

Mostly copy-paste from [https://github.com/facebookresearch/mae/blob/main/models\\_mae.py](https://github.com/facebookresearch/mae/blob/main/models_mae.py)

```
class easycv.models.backbones.mae_vit_transformer.MaskedAutoencoderViT(img_size=224,
 patch_size=16,
 in_chans=3,
 embed_dim=1024,
 depth=24,
 num_heads=16,
 mlp_ratio=4.0,
 norm_layer=functools.partial(<class
 'torch.nn.modules.normalization.LayerNorm'
 eps=1e-06))
```

Bases: `torch.nn.modules.module.Module`

**Masked Autoencoder with VisionTransformer backbone.** MaskedAutoencoderViT is mostly same as vit\_tranformer\_dynamic, but with a random\_masking func. MaskedAutoencoderViT model can be loaded by vit\_tranformer\_dynamic.

#### **Parameters**

- **img\_size** (*int*) – input image size
- **patch\_size** (*int*) – patch size
- **in\_chans** (*int*) – input image channels
- **embed\_dim** (*int*) – feature dimensions
- **depth** (*int*) – number of encoder layers
- **num\_heads** (*int*) – Parallel attention heads
- **mlp\_ratio** (*float*) – mlp ratio
- **norm\_layer** – type of normalization layer

```
__init__(img_size=224, patch_size=16, in_chans=3, embed_dim=1024, depth=24, num_heads=16,
 mlp_ratio=4.0, norm_layer=functools.partial(<class
 'torch.nn.modules.normalization.LayerNorm'>, eps=1e-06))
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**random\_masking**(*x, mask\_ratio*)

Perform per-sample random masking by per-sample shuffling. Per-sample shuffling is done by argsort random noise. *x*: [N, L, D], sequence

**forward**(*x, mask\_ratio*)

Defines the computation performed at every call.

Should be overridden by all subclasses.



---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

### easycv.models.backbones.mnasnet module

This model is taken from the official PyTorch model zoo. - torchvision.models.mnasnet.py on 31th Aug, 2019

**class** easycv.models.backbones.mnasnet.**MNASNet**(alpha, num\_classes=0, dropout=0.2)

Bases: torch.nn.modules.module.Module

MNASNet, as described in <https://arxiv.org/pdf/1807.11626.pdf>. >>> model = MNASNet(1000, 1.0) >>> x = torch.rand(1, 3, 224, 224) >>> y = model(x) >>> y.dim() 1 >>> y.nelement() 1000

**\_\_init\_\_**(alpha, num\_classes=0, dropout=0.2)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**init\_weights**(pretrained=None)

**training:** bool

### easycv.models.backbones.mobilenetv2 module

This model is taken from the official PyTorch model zoo. - torchvision.models.mobilenet.py on 31th Aug, 2019

**class** easycv.models.backbones.mobilenetv2.**MobileNetV2**(num\_classes=0, width\_multi=1.0, inverted\_residual\_setting=None, round\_nearest=8)

Bases: torch.nn.modules.module.Module

**\_\_init\_\_**(num\_classes=0, width\_multi=1.0, inverted\_residual\_setting=None, round\_nearest=8)

MobileNet V2 main class :param num\_classes: Number of classes :type num\_classes: int :param width\_multi: Width multiplier - adjusts number of channels in each layer by this amount :type width\_multi: float :param inverted\_residual\_setting: Network structure :param round\_nearest: Round the number of channels in each layer to be a multiple of this number :type round\_nearest: int :param Set to 1 to turn off rounding:

**init\_weights**(pretrained=None)

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

### **easycv.models.backbones.network\_blocks module**

**class** easycv.models.backbones.network\_blocks.**SiLU**(*inplace=True*)

Bases: torch.nn.modules.module.Module

export-friendly inplace version of nn.SiLU()

**\_\_init\_\_**(*inplace=True*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**static forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**class** easycv.models.backbones.network\_blocks.**HSiLU**(*inplace=True*)

Bases: torch.nn.modules.module.Module

export-friendly inplace version of nn.SiLU() hardsigmoid is better than sigmoid when used for edge model

**\_\_init\_\_**(*inplace=True*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**static forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

easycv.models.backbones.network\_blocks.**get\_activation**(*name='silu', inplace=True*)

**class** easycv.models.backbones.network\_blocks.**BaseConv**(*in\_channels, out\_channels, ksize, stride, groups=1, bias=False, act='silu'*)

Bases: torch.nn.modules.module.Module

A Conv2d -> Batchnorm -> silu/leaky relu block

**\_\_init\_\_**(*in\_channels, out\_channels, ksize, stride, groups=1, bias=False, act='silu'*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**fuseforward(x)**

**training:** bool

```
class easycv.models.backbones.network_blocks.DWConv(in_channels, out_channels, ksize, stride=1,
 act='silu')
```

Bases: torch.nn.modules.module.Module

Depthwise Conv + Conv

```
__init__(in_channels, out_channels, ksize, stride=1, act='silu')
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.network_blocks.Bottleneck(in_channels, out_channels, shortcut=True,
 expansion=0.5, depthwise=False,
 act='silu')
```

Bases: torch.nn.modules.module.Module

```
__init__(in_channels, out_channels, shortcut=True, expansion=0.5, depthwise=False, act='silu')
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.network_blocks.ResLayer(in_channels: int)
```

Bases: torch.nn.modules.module.Module

Residual layer with *in\_channels* inputs.

**\_\_init\_\_**(*in\_channels: int*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** easycv.models.backbones.network\_blocks.**SPPBottleneck**(*in\_channels, out\_channels,*  
*kernel\_sizes=(5, 9, 13),*  
*activation='silu'*)

Bases: torch.nn.modules.module.Module

Spatial pyramid pooling layer used in YOLOv3-SPP

**\_\_init\_\_**(*in\_channels, out\_channels, kernel\_sizes=(5, 9, 13), activation='silu'*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** easycv.models.backbones.network\_blocks.**CSPLayer**(*in\_channels, out\_channels, n=1,*  
*shortcut=True, expansion=0.5,*  
*depthwise=False, act='silu'*)

Bases: torch.nn.modules.module.Module

CSP Bottleneck with 3 convolutions

**\_\_init\_\_**(*in\_channels, out\_channels, n=1, shortcut=True, expansion=0.5, depthwise=False, act='silu'*)

#### Parameters

- **in\_channels** (*int*) – input channels.
- **out\_channels** (*int*) – output channels.
- **n** (*int*) – number of Bottlenecks. Default value: 1.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `easycv.models.backbones.network_blocks.Focus`(*in\_channels*, *out\_channels*, *ksize=1*, *stride=1*, *act='silu'*)

Bases: `torch.nn.modules.module.Module`

Focus width and height information into channel space.

**\_\_init\_\_**(*in\_channels*, *out\_channels*, *ksize=1*, *stride=1*, *act='silu'*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

### `easycv.models.backbones.pytorch_image_models_wrapper` module

**class** `easycv.models.backbones.pytorch_image_models_wrapper.PytorchImageModelWrapper`(*model\_name='resnet50'*, *pre-trained=False*, *checkpoint\_path=None*, *scriptable=None*, *exportable=None*, *no\_jit=None*, *\*\*kwargs*)

Bases: `torch.nn.modules.module.Module`

Support Backbones From `pytorch-image-models`.

The PyTorch community has lots of awesome contributions for image models. PyTorch Image Models (timm) is a collection of image models, aim to pull together a wide variety of SOTA models with ability to reproduce ImageNet training results.

Model pages can be found at <https://rwightman.github.io/pytorch-image-models/models/>

References: <https://github.com/rwightman/pytorch-image-models>

**\_\_init\_\_**(*model\_name='resnet50'*, *pretrained=False*, *checkpoint\_path=None*, *scriptable=None*, *exportable=None*, *no\_jit=None*, *\*\*kwargs*)

Init `PytorchImageModelWrapper` by `timm.create_model` :param *model\_name*: name of model to instantiate :type *model\_name*: str :param *pretrained*: load pretrained ImageNet-1k weights if true :type

pretrained: bool :param checkpoint\_path: path of checkpoint to load after model is initialized :type checkpoint\_path: str :param scriptable: set layer config so that model is jit scriptable (not working for all models yet) :type scriptable: bool :param exportable: set layer config so that model is traceable / ONNX exportable (not fully impl/obeyed yet) :type exportable: bool :param no\_jit: set layer config so that model doesn't utilize jit scripted layers (so far activations only) :type no\_jit: bool

**init\_weights**(*pretrained=None*)

**training:** bool

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

## easycv.models.backbones.resnest module

ResNet variants

**class** easycv.models.backbones.resnest.**SplAtConv2d**(*in\_channels, channels, kernel\_size, stride=(1, 1), padding=(0, 0), dilation=(1, 1), groups=1, bias=True, radix=2, reduction\_factor=4, rectify=False, rectify\_avg=False, norm\_layer=None, dropblock\_prob=0.0, \*\*kwargs*)

Bases: torch.nn.modules.module.Module

Split-Attention Conv2d

**\_\_init\_\_**(*in\_channels, channels, kernel\_size, stride=(1, 1), padding=(0, 0), dilation=(1, 1), groups=1, bias=True, radix=2, reduction\_factor=4, rectify=False, rectify\_avg=False, norm\_layer=None, dropblock\_prob=0.0, \*\*kwargs*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**class** easycv.models.backbones.resnest.**rSoftMax**(*radix, cardinality*)

Bases: torch.nn.modules.module.Module

**\_\_init\_\_**(*radix, cardinality*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `easycv.models.backbones.resnest.DropBlock2D(*args, **kwargs)`

Bases: `object`

**\_\_init\_\_**(\*args, \*\*kwargs)

Initialize self. See `help(type(self))` for accurate signature.

**class** `easycv.models.backbones.resnest.GlobalAvgPool2d`

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**()

Global average pooling over the input's spatial dimensions

**forward**(inputs)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `easycv.models.backbones.resnest.Bottleneck(inplanes, planes, stride=1, downsample=None, radix=1, cardinality=1, bottleneck_width=64, avd=False, avd_first=False, dilation=1, is_first=False, rectified_conv=False, rectify_avg=False, norm_layer=None, dropblock_prob=0.0, last_gamma=False)`

Bases: `torch.nn.modules.module.Module`

ResNet Bottleneck

**expansion** = 4

**\_\_init\_\_**(inplanes, planes, stride=1, downsample=None, radix=1, cardinality=1, bottleneck\_width=64, avd=False, avd\_first=False, dilation=1, is\_first=False, rectified\_conv=False, rectify\_avg=False, norm\_layer=None, dropblock\_prob=0.0, last\_gamma=False)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.resnest.ResNeSt(depth=None, block=<class
 'easycv.models.backbones.resnest.Bottleneck'>,
 layers=[3, 4, 6, 3], radix=2, groups=1,
 bottleneck_width=64, num_classes=0, dilated=False,
 dilation=1, deep_stem=True, stem_width=32,
 avg_down=True, rectified_conv=False,
 rectify_avg=False, avd=False, avd_first=False,
 final_drop=0.0, dropblock_prob=0, last_gamma=False,
 norm_layer=<class
 'torch.nn.modules.batchnorm.BatchNorm2d'>)
```

Bases: torch.nn.modules.module.Module

ResNet Variants

#### Parameters

- **block** ([Block](#)) – Class for the residual block. Options are BasicBlockV1, BottleneckV1.
- **layers** (*list of int*) – Numbers of layers in each block
- **classes** (*int*, *default 1000*) – Number of classification classes.
- **dilated** (*bool*, *default False*) – Applying dilation strategy to pretrained ResNet yielding a stride-8 model, typically used in Semantic Segmentation.
- **norm\_layer** (*object*) – Normalization layer used in backbone network (default: mxnet.gluon.nn.BatchNorm; for Synchronized Cross-GPU BatchNormalization).
- **Reference** –
  - He, Kaiming, et al. “Deep residual learning for image recognition.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
  - Yu, Fisher, and Vladlen Koltun. “Multi-scale context aggregation by dilated convolutions.”

```
arch_settings = {50: ((3, 4, 6, 3), 32), 101: ((3, 4, 23, 3), 64), 200: ((3, 24,
36, 3), 64), 269: ((3, 30, 48, 8), 64)}
```

```
__init__(depth=None, block=<class 'easycv.models.backbones.resnest.Bottleneck'>, layers=[3, 4, 6, 3],
 radix=2, groups=1, bottleneck_width=64, num_classes=0, dilated=False, dilation=1,
 deep_stem=True, stem_width=32, avg_down=True, rectified_conv=False, rectify_avg=False,
 avd=False, avd_first=False, final_drop=0.0, dropblock_prob=0, last_gamma=False,
 norm_layer=<class 'torch.nn.modules.batchnorm.BatchNorm2d'>)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**training:** bool

**init\_weights**(*pretrained=None*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---



**easycv.models.backbones.resnet module**

```
class easycv.models.backbones.resnet.BasicBlock(inplanes, planes, stride=1, dilation=1,
 downsample=None, style='pytorch', with_cp=False,
 conv_cfg=None, norm_cfg={'type': 'BN'},
 frelu=False)
```

Bases: torch.nn.modules.module.Module

**expansion** = 1

```
__init__(inplanes, planes, stride=1, dilation=1, downsample=None, style='pytorch', with_cp=False,
 conv_cfg=None, norm_cfg={'type': 'BN'}, frelu=False)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**property norm1**

**property norm2**

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.resnet.Bottleneck(inplanes, planes, stride=1, dilation=1,
 downsample=None, style='pytorch', with_cp=False,
 conv_cfg=None, norm_cfg={'type': 'BN'},
 frelu=False)
```

Bases: torch.nn.modules.module.Module

**expansion** = 4

```
__init__(inplanes, planes, stride=1, dilation=1, downsample=None, style='pytorch', with_cp=False,
 conv_cfg=None, norm_cfg={'type': 'BN'}, frelu=False)
```

Bottleneck block for ResNet. If style is “pytorch”, the stride-two layer is the 3x3 conv layer, if it is “caffe”, the stride-two layer is the first 1x1 conv layer.

**property norm1**

**property norm2**

**property norm3**

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
easycv.models.backbones.resnet.make_res_layer(block, inplanes, planes, blocks, stride=1, dilation=1,
 style='pytorch', with_cp=False, conv_cfg=None,
 norm_cfg={'type': 'BN'}, frelu=False)
```

```
class easycv.models.backbones.resnet.ResNet(depth, in_channels=3, num_stages=4, strides=(1, 2, 2, 2),
 dilations=(1, 1, 1, 1), out_indices=(0, 1, 2, 3, 4),
 style='pytorch', num_classes=0, frozen_stages=-1,
 conv_cfg=None, norm_cfg={'requires_grad': True, 'type':
 'BN'}, norm_eval=False, with_cp=False, frelu=False,
 original_inplanes=64, zero_init_residual=False)
```

Bases: torch.nn.modules.module.Module

ResNet backbone.

#### Parameters

- **depth** (*int*) – Depth of resnet, from {18, 34, 50, 101, 152}.
- **in\_channels** (*int*) – Number of input image channels. Normally 3.
- **num\_stages** (*int*) – Resnet stages, normally 4.
- **strides** (*Sequence[int]*) – Strides of the first block of each stage.
- **dilations** (*Sequence[int]*) – Dilation of each stage.
- **out\_indices** (*Sequence[int]*) – Output from which stages.
- **style** (*str*) – *pytorch* or *caffe*. If set to “pytorch”, the stride-two layer is the 3x3 conv layer, otherwise the stride-two layer is the first 1x1 conv layer.
- **frozen\_stages** (*int*) – Stages to be frozen (stop grad and set eval mode). -1 means not freezing any parameters.
- **norm\_cfg** (*dict*) – dictionary to construct and config norm layer.
- **norm\_eval** (*bool*) – Whether to set norm layers to eval mode, namely, freeze running stats (mean and var). Note: Effect on Batch Norm and its variants only.
- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed.
- **original\_inplanes** – start channel for first block, default=64
- **zero\_init\_residual** (*bool*) – whether to use zero init for last norm layer in resblocks to let them behave as identity.

#### Example

```
>>> from easycv.models import ResNet
>>> import torch
>>> self = ResNet(depth=18)
>>> self.eval()
>>> inputs = torch.rand(1, 3, 32, 32)
>>> level_outputs = self.forward(inputs)
>>> for level_out in level_outputs:
... print(tuple(level_out.shape))
(1, 64, 8, 8)
(1, 128, 4, 4)
(1, 256, 2, 2)
(1, 512, 1, 1)
```

```

arch_settings = {10: (<class 'easycv.models.backbones.resnet.BasicBlock'>, (1, 1,
1, 1)), 18: (<class 'easycv.models.backbones.resnet.BasicBlock'>, (2, 2, 2, 2)),
34: (<class 'easycv.models.backbones.resnet.BasicBlock'>, (3, 4, 6, 3)), 50:
(<class 'easycv.models.backbones.resnet.Bottleneck'>, (3, 4, 6, 3)), 101: (<class
'easycv.models.backbones.resnet.Bottleneck'>, (3, 4, 23, 3)), 152: (<class
'easycv.models.backbones.resnet.Bottleneck'>, (3, 8, 36, 3))}

__init__(depth, in_channels=3, num_stages=4, strides=(1, 2, 2, 2), dilations=(1, 1, 1, 1), out_indices=(0, 1,
2, 3, 4), style='pytorch', num_classes=0, frozen_stages=- 1, conv_cfg=None,
norm_cfg={'requires_grad': True, 'type': 'BN'}, norm_eval=False, with_cp=False, frelu=False,
original_inplanes=64, zero_init_residual=False)
Initializes internal Module state, shared by both nn.Module and ScriptModule.

```

**property** norm1

**init\_weights**(pretrained=None)

**training:** bool

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**train**(mode=True)

Sets the module in training mode.

This has any effect only on certain modules. See documentations of particular modules for details of their behaviors in training/evaluation mode, if they are affected, e.g. Dropout, BatchNorm, etc.

**Parameters** mode (bool) – whether to set training mode (True) or evaluation mode (False).

Default: True.

**Returns** self

**Return type** Module

## easycv.models.backbones.resnet\_jit module

```

class easycv.models.backbones.resnet_jit.BasicBlock(inplanes, planes, stride=1, dilation=1,
downsample=None, style='pytorch',
with_cp=False, conv_cfg=None,
norm_cfg={'type': 'BN'})

```

Bases: torch.nn.modules.module.Module

**expansion** = 1

```

__init__(inplanes, planes, stride=1, dilation=1, downsample=None, style='pytorch', with_cp=False,
conv_cfg=None, norm_cfg={'type': 'BN'})

```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**property** norm1

**property** norm2

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.resnet_jit.Bottleneck(inplanes, planes, stride=1, dilation=1,
 downsample=None, style='pytorch',
 with_cp=False, conv_cfg=None,
 norm_cfg={'type': 'BN'})
```

Bases: torch.nn.modules.module.Module

**expansion** = 4

```
__init__(inplanes, planes, stride=1, dilation=1, downsample=None, style='pytorch', with_cp=False,
 conv_cfg=None, norm_cfg={'type': 'BN'})
```

Bottleneck block for ResNet. If style is “pytorch”, the stride-two layer is the 3x3 conv layer, if it is “caffe”, the stride-two layer is the first 1x1 conv layer.

**property norm1**

**property norm2**

**property norm3**

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
easycv.models.backbones.resnet_jit.make_res_layer(block, inplanes, planes, blocks, stride=1,
 dilation=1, style='pytorch', with_cp=False,
 conv_cfg=None, norm_cfg={'type': 'BN'})
```

```
class easycv.models.backbones.resnet_jit.ResNetJIT(depth, in_channels=3, num_stages=4, strides=(1,
2, 2, 2), dilations=(1, 1, 1, 1), out_indices=(0, 1, 2, 3, 4), style='pytorch', frozen_stages=-1,
conv_cfg=None, norm_cfg={'requires_grad':
True, 'type': 'BN'}, norm_eval=False,
with_cp=False, zero_init_residual=False)
```

Bases: torch.nn.modules.module.Module

ResNet backbone.

**Parameters**

- **depth** (*int*) – Depth of resnet, from {18, 34, 50, 101, 152}.
- **in\_channels** (*int*) – Number of input image channels. Normally 3.

- **num\_stages** (*int*) – Resnet stages, normally 4.
- **strides** (*Sequence[int]*) – Strides of the first block of each stage.
- **dilations** (*Sequence[int]*) – Dilation of each stage.
- **out\_indices** (*Sequence[int]*) – Output from which stages.
- **style** (*str*) – *pytorch* or *caffe*. If set to “pytorch”, the stride-two layer is the 3x3 conv layer, otherwise the stride-two layer is the first 1x1 conv layer.
- **frozen\_stages** (*int*) – Stages to be frozen (stop grad and set eval mode). -1 means not freezing any parameters.
- **norm\_cfg** (*dict*) – dictionary to construct and config norm layer.
- **norm\_eval** (*bool*) – Whether to set norm layers to eval mode, namely, freeze running stats (mean and var). Note: Effect on Batch Norm and its variants only.
- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed.
- **zero\_init\_residual** (*bool*) – whether to use zero init for last norm layer in resblocks to let them behave as identity.

### Example

```
>>> from easycv.models import ResNet
>>> import torch
>>> self = ResNet(depth=18)
>>> self.eval()
>>> inputs = torch.rand(1, 3, 32, 32)
>>> level_outputs = self.forward(inputs)
>>> for level_out in level_outputs:
... print(tuple(level_out.shape))
(1, 64, 8, 8)
(1, 128, 4, 4)
(1, 256, 2, 2)
(1, 512, 1, 1)
```

```
arch_settings = {18: (<class 'easycv.models.backbones.resnet_jit.BasicBlock'>, (2, 2, 2, 2)), 34: (<class 'easycv.models.backbones.resnet_jit.BasicBlock'>, (3, 4, 6, 3)), 50: (<class 'easycv.models.backbones.resnet_jit.Bottleneck'>, (3, 4, 6, 3)), 101: (<class 'easycv.models.backbones.resnet_jit.Bottleneck'>, (3, 4, 23, 3)), 152: (<class 'easycv.models.backbones.resnet_jit.Bottleneck'>, (3, 8, 36, 3))}
```

```
__init__(depth, in_channels=3, num_stages=4, strides=(1, 2, 2, 2), dilations=(1, 1, 1, 1), out_indices=(0, 1, 2, 3, 4), style='pytorch', frozen_stages=-1, conv_cfg=None, norm_cfg={'requires_grad': True, 'type': 'BN'}, norm_eval=False, with_cp=False, zero_init_residual=False)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**property norm1**

**init\_weights**(*pretrained=None*)

**training**: *bool*

**forward**(*x: torch.Tensor*) → List[torch.Tensor]

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**train**(*mode=True*)

Sets the module in training mode.

This has any effect only on certain modules. See documentations of particular modules for details of their behaviors in training/evaluation mode, if they are affected, e.g. `Dropout`, `BatchNorm`, etc.

**Parameters** *mode* (*bool*) – whether to set training mode (`True`) or evaluation mode (`False`).

Default: `True`.

**Returns** *self*

**Return type** `Module`

### `easycv.models.backbones.resnext` module

**class** `easycv.models.backbones.resnext.Bottleneck`(*inplanes, planes, groups=1, base\_width=4, \*\*kwargs*)

Bases: `easycv.models.backbones.resnet.Bottleneck`

**\_\_init\_\_**(*inplanes, planes, groups=1, base\_width=4, \*\*kwargs*)

Bottleneck block for ResNeXt. If style is “pytorch”, the stride-two layer is the 3x3 conv layer, if it is “caffe”, the stride-two layer is the first 1x1 conv layer.

**training:** *bool*

`easycv.models.backbones.resnext.make_res_layer`(*block, inplanes, planes, blocks, stride=1, dilation=1, groups=1, base\_width=4, style='pytorch', with\_cp=False, conv\_cfg=None, norm\_cfg='type': 'BN'*)

**class** `easycv.models.backbones.resnext.ResNeXt`(*groups=1, base\_width=4, \*\*kwargs*)

Bases: `easycv.models.backbones.resnet.ResNet`

ResNeXt backbone.

**Parameters**

- **depth** (*int*) – Depth of resnet, from {18, 34, 50, 101, 152}.
- **in\_channels** (*int*) – Number of input image channels. Normally 3.
- **num\_stages** (*int*) – Resnet stages, normally 4.
- **groups** (*int*) – Group of resnext.
- **base\_width** (*int*) – Base width of resnext.
- **strides** (*Sequence[int]*) – Strides of the first block of each stage.
- **dilations** (*Sequence[int]*) – Dilation of each stage.
- **out\_indices** (*Sequence[int]*) – Output from which stages.
- **style** (*str*) – *pytorch* or *caffe*. If set to “pytorch”, the stride-two layer is the 3x3 conv layer, otherwise the stride-two layer is the first 1x1 conv layer.

- **frozen\_stages** (*int*) – Stages to be frozen (all param fixed). -1 means not freezing any parameters.
- **norm\_cfg** (*dict*) – dictionary to construct and config norm layer.
- **norm\_eval** (*bool*) – Whether to set norm layers to eval mode, namely, freeze running stats (mean and var). Note: Effect on Batch Norm and its variants only.
- **with\_cp** (*bool*) – Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed.
- **zero\_init\_residual** (*bool*) – whether to use zero init for last norm layer in resblocks to let them behave as identity.

### Example

```
>>> from easycv.models import ResNeXt
>>> import torch
>>> self = ResNeXt(depth=50)
>>> self.eval()
>>> inputs = torch.rand(1, 3, 32, 32)
>>> level_outputs = self.forward(inputs)
>>> for level_out in level_outputs:
... print(tuple(level_out.shape))
(1, 256, 8, 8)
(1, 512, 4, 4)
(1, 1024, 2, 2)
(1, 2048, 1, 1)
```

```
arch_settings = {50: (<class 'easycv.models.backbones.resnext.Bottleneck'>, (3, 4, 6, 3)), 101: (<class 'easycv.models.backbones.resnext.Bottleneck'>, (3, 4, 23, 3)), 152: (<class 'easycv.models.backbones.resnext.Bottleneck'>, (3, 8, 36, 3))}
```

```
__init__(groups=1, base_width=4, **kwargs)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
training: bool
```

### easycv.models.backbones.shuffle\_transformer module

```
class easycv.models.backbones.shuffle_transformer.Mlp(in_features, hidden_features=None,
out_features=None, act_layer=<class 'torch.nn.modules.activation.ReLU6'>,
drop=0.0, stride=False)
```

Bases: torch.nn.modules.module.Module

```
__init__(in_features, hidden_features=None, out_features=None, act_layer=<class 'torch.nn.modules.activation.ReLU6'>, drop=0.0, stride=False)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.shuffle_transformer.Attention(dim, num_heads, window_size=1,
 shuffle=False, qkv_bias=False,
 qk_scale=None, attn_drop=0.0,
 proj_drop=0.0,
 relative_pos_embedding=False)
```

Bases: torch.nn.modules.module.Module

```
__init__(dim, num_heads, window_size=1, shuffle=False, qkv_bias=False, qk_scale=None, attn_drop=0.0,
 proj_drop=0.0, relative_pos_embedding=False)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.shuffle_transformer.Block(dim, out_dim, num_heads, window_size=1,
 shuffle=False, mlp_ratio=4.0,
 qkv_bias=False, qk_scale=None,
 drop=0.0, attn_drop=0.0, drop_path=0.0,
 act_layer=<class
 'torch.nn.modules.activation.ReLU6'>,
 norm_layer=<class
 'torch.nn.modules.batchnorm.BatchNorm2d'>,
 stride=False,
 relative_pos_embedding=False)
```

Bases: torch.nn.modules.module.Module

```
__init__(dim, out_dim, num_heads, window_size=1, shuffle=False, mlp_ratio=4.0, qkv_bias=False,
 qk_scale=None, drop=0.0, attn_drop=0.0, drop_path=0.0, act_layer=<class
 'torch.nn.modules.activation.ReLU6'>, norm_layer=<class
 'torch.nn.modules.batchnorm.BatchNorm2d'>, stride=False, relative_pos_embedding=False)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---



**training:** bool

```
class easycv.models.backbones.shuffle_transformer.PatchMerging(dim, out_dim, norm_layer=<class
 'torch.nn.modules.batchnorm.BatchNorm2d'>)
```

Bases: torch.nn.modules.module.Module

```
__init__(dim, out_dim, norm_layer=<class 'torch.nn.modules.batchnorm.BatchNorm2d'>)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**extra\_repr**() → str

Set the extra representation of the module

To print customized extra information, you should re-implement this method in your own modules. Both single-line and multi-line strings are acceptable.

**training:** bool

```
class easycv.models.backbones.shuffle_transformer.StageModule(layers, dim, out_dim, num_heads,
 window_size=1, shuffle=True,
 mlp_ratio=4.0, qkv_bias=False,
 qk_scale=None, drop=0.0,
 attn_drop=0.0, drop_path=0.0,
 act_layer=<class
 'torch.nn.modules.activation.ReLU6'>,
 norm_layer=<class
 'torch.nn.modules.batchnorm.BatchNorm2d'>,
 relative_pos_embedding=False)
```

Bases: torch.nn.modules.module.Module

```
__init__(layers, dim, out_dim, num_heads, window_size=1, shuffle=True, mlp_ratio=4.0, qkv_bias=False,
 qk_scale=None, drop=0.0, attn_drop=0.0, drop_path=0.0, act_layer=<class
 'torch.nn.modules.activation.ReLU6'>, norm_layer=<class
 'torch.nn.modules.batchnorm.BatchNorm2d'>, relative_pos_embedding=False)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.shuffle_transformer.PatchEmbedding(inter_channel=32,
 out_channels=48)
```

Bases: torch.nn.modules.module.Module

```
__init__(inter_channel=32, out_channels=48)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.shuffle_transformer.ShuffleTransformer(img_size=224,
 in_chans=3,
 num_classes=1000,
 token_dim=32,
 embed_dim=96,
 mlp_ratio=4.0, layers=[2,
 2, 6, 2], num_heads=[3, 6,
 12, 24], rela-
 tive_pos_embedding=True,
 shuffle=True,
 window_size=7,
 qkv_bias=True,
 qk_scale=None,
 drop_rate=0.0,
 attn_drop_rate=0.0,
 drop_path_rate=0.0,
 has_pos_embed=False,
 **kwargs)
```

Bases: torch.nn.modules.module.Module

```
__init__(img_size=224, in_chans=3, num_classes=1000, token_dim=32, embed_dim=96, mlp_ratio=4.0,
 layers=[2, 2, 6, 2], num_heads=[3, 6, 12, 24], relative_pos_embedding=True, shuffle=True,
 window_size=7, qkv_bias=True, qk_scale=None, drop_rate=0.0, attn_drop_rate=0.0,
 drop_path_rate=0.0, has_pos_embed=False, **kwargs)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
init_weights(pretrained=None)
```

```
no_weight_decay()
```

```
no_weight_decay_keywords()
```

```
get_classifier()
```

```
reset_classifier(num_classes, global_pool="")
```

```
forward_features(x)
```

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

`easycv.models.backbones.shuffle_transformer.shuffletrans_base_p4_w7_224(pretrained=False, **kwargs)`

`easycv.models.backbones.shuffle_transformer.shuffletrans_small_p4_w7_224(pretrained=False, **kwargs)`

`easycv.models.backbones.shuffle_transformer.shuffletrans_tiny_p4_w7_224(pretrained=False, **kwargs)`

### `easycv.models.backbones.swin_transformer_dynamic` module

Borrow this code from [https://github.com/microsoft/esvit/blob/main/models/swin\\_transformer.py](https://github.com/microsoft/esvit/blob/main/models/swin_transformer.py) To support dynamic swin-transformer for ssl!

```
class easycv.models.backbones.swin_transformer_dynamic.Mlp(in_features, hidden_features=None,
 out_features=None, act_layer=<class
 'torch.nn.modules.activation.GELU'>,
 drop=0.0)
```

Bases: `torch.nn.modules.module.Module`

```
__init__(in_features, hidden_features=None, out_features=None, act_layer=<class
 'torch.nn.modules.activation.GELU'>, drop=0.0)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

`easycv.models.backbones.swin_transformer_dynamic.window_partition(x, window_size)`

#### Parameters

- **x** – (B, H, W, C)
- **window\_size** (*int*) – window size

**Returns** (num\_windows\*B, window\_size, window\_size, C)

**Return type** windows

`easycv.models.backbones.swin_transformer_dynamic.window_reverse(windows, window_size, H, W)`

#### Parameters

- **windows** – (num\_windows\*B, window\_size, window\_size, C)
- **window\_size** (*int*) – Window size
- **H** (*int*) – Height of image
- **W** (*int*) – Width of image

**Returns** (B, H, W, C)

**Return type** *x*

```
class easycv.models.backbones.swin_transformer_dynamic.WindowAttention(dim, window_size,
 num_heads,
 qkv_bias=True,
 qk_scale=None,
 attn_drop=0.0,
 proj_drop=0.0)
```

Bases: torch.nn.modules.module.Module

Window based multi-head self attention (W-MSA) module with relative position bias. It supports both of shifted and non-shifted window.

#### Parameters

- **dim** (*int*) – Number of input channels.
- **window\_size** (*tuple[int]*) – The height and width of the window.
- **num\_heads** (*int*) – Number of attention heads.
- **qkv\_bias** (*bool*, *optional*) – If True, add a learnable bias to query, key, value. Default: True
- **qk\_scale** (*float | None*, *optional*) – Override default qk scale of head\_dim \*\* -0.5 if set
- **attn\_drop** (*float*, *optional*) – Dropout ratio of attention weight. Default: 0.0
- **proj\_drop** (*float*, *optional*) – Dropout ratio of output. Default: 0.0

**\_\_init\_\_** (*dim*, *window\_size*, *num\_heads*, *qkv\_bias=True*, *qk\_scale=None*, *attn\_drop=0.0*, *proj\_drop=0.0*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*, *mask=None*)

#### Parameters

- **x** – input features with shape of (num\_windows\*B, N, C)
- **mask** – (0/-inf) mask with shape of (num\_windows, Wh\*Ww, Wh\*Ww) or None

**extra\_repr** () → str

Set the extra representation of the module

To print customized extra information, you should re-implement this method in your own modules. Both single-line and multi-line strings are acceptable.

**flops** (*N*)

**static compute\_macs** (*module*, *input*, *output*)

**training:** bool

```
class easycv.models.backbones.swin_transformer_dynamic.SwinTransformerBlock(dim,
 input_resolution,
 num_heads,
 window_size=7,
 shift_size=0,
 mlp_ratio=4.0,
 qkv_bias=True,
 qk_scale=None,
 drop=0.0,
 attn_drop=0.0,
 drop_path=0.0,
 act_layer=<class
 'torch.nn.modules.activation.GELU'>,
 norm_layer=<class
 'torch.nn.modules.normalization.La
```

Bases: torch.nn.modules.module.Module

Swin Transformer Block.

#### Parameters

- **dim** (*int*) – Number of input channels.
- **input\_resolution** (*tuple[int]*) – Input resolution.
- **num\_heads** (*int*) – Number of attention heads.
- **window\_size** (*int*) – Window size.
- **shift\_size** (*int*) – Shift size for SW-MSA.
- **mlp\_ratio** (*float*) – Ratio of mlp hidden dim to embedding dim.
- **qkv\_bias** (*bool*, *optional*) – If True, add a learnable bias to query, key, value. Default: True
- **qk\_scale** (*float | None*, *optional*) – Override default qk scale of head\_dim \*\* -0.5 if set.
- **drop** (*float*, *optional*) – Dropout rate. Default: 0.0
- **attn\_drop** (*float*, *optional*) – Attention dropout rate. Default: 0.0
- **drop\_path** (*float*, *optional*) – Stochastic depth rate. Default: 0.0
- **act\_layer** (*nn.Module*, *optional*) – Activation layer. Default: nn.GELU
- **norm\_layer** (*nn.Module*, *optional*) – Normalization layer. Default: nn.LayerNorm

```
__init__(dim, input_resolution, num_heads, window_size=7, shift_size=0, mlp_ratio=4.0, qkv_bias=True,
 qk_scale=None, drop=0.0, attn_drop=0.0, drop_path=0.0, act_layer=<class
 'torch.nn.modules.activation.GELU'>, norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
create_attn_mask(H, W)
```

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks

while the latter silently ignores them.

---

**extra\_repr()** → str

Set the extra representation of the module

To print customized extra information, you should re-implement this method in your own modules. Both single-line and multi-line strings are acceptable.

**flops()**

**training:** bool

```
class easycv.models.backbones.swin_transformer_dynamic.PatchMerging(input_resolution, dim,
 norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>)
```

Bases: torch.nn.modules.module.Module

Patch Merging Layer.

**Parameters**

- **input\_resolution** (tuple[int]) – Resolution of input feature.
- **dim** (int) – Number of input channels.
- **norm\_layer** (nn.Module, optional) – Normalization layer. Default: nn.LayerNorm

```
__init__(input_resolution, dim, norm_layer=<class 'torch.nn.modules.normalization.LayerNorm'>)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(x)

Forward function. :param x: Input feature, tensor size (B, H\*W, C). :param H: Spatial resolution of the input feature. :param W: Spatial resolution of the input feature.

**extra\_repr()** → str

Set the extra representation of the module

To print customized extra information, you should re-implement this method in your own modules. Both single-line and multi-line strings are acceptable.

**flops()**

**training:** bool

```
class easycv.models.backbones.swin_transformer_dynamic.BasicLayer(dim, input_resolution, depth,
 num_heads, window_size,
 mlp_ratio=4.0,
 qkv_bias=True,
 qk_scale=None, drop=0.0,
 attn_drop=0.0,
 drop_path=0.0,
 norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>,
 downsample=None)
```

Bases: torch.nn.modules.module.Module

A basic Swin Transformer layer for one stage.

**Parameters**

- **dim** (int) – Number of input channels.
- **input\_resolution** (tuple[int]) – Input resolution.
- **depth** (int) – Number of blocks.

- **num\_heads** (*int*) – Number of attention heads.
- **window\_size** (*int*) – Window size.
- **mlp\_ratio** (*float*) – Ratio of mlp hidden dim to embedding dim.
- **qkv\_bias** (*bool*, *optional*) – If True, add a learnable bias to query, key, value. Default: True
- **qk\_scale** (*float* | *None*, *optional*) – Override default qk scale of head\_dim \*\* -0.5 if set.
- **drop** (*float*, *optional*) – Dropout rate. Default: 0.0
- **attn\_drop** (*float*, *optional*) – Attention dropout rate. Default: 0.0
- **drop\_path** (*float* | *tuple[float]*, *optional*) – Stochastic depth rate. Default: 0.0
- **norm\_layer** (*nn.Module*, *optional*) – Normalization layer. Default: nn.LayerNorm
- **downsample** (*nn.Module* | *None*, *optional*) – Downsample layer at the end of the layer. Default: None

**\_\_init\_\_** (*dim*, *input\_resolution*, *depth*, *num\_heads*, *window\_size*, *mlp\_ratio*=4.0, *qkv\_bias*=True, *qk\_scale*=None, *drop*=0.0, *attn\_drop*=0.0, *drop\_path*=0.0, *norm\_layer*=<class 'torch.nn.modules.normalization.LayerNorm'>, *downsample*=None)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**forward\_with\_features** (*x*)

**forward\_with\_attention** (*x*)

**extra\_repr** () → str

Set the extra representation of the module

To print customized extra information, you should re-implement this method in your own modules. Both single-line and multi-line strings are acceptable.

**flops** ()

**training:** bool

**class** easycv.models.backbones.swin\_transformer\_dynamic.**PatchEmbed** (*img\_size*=224, *patch\_size*=16, *in\_chans*=3, *embed\_dim*=768, *norm\_layer*=None)

Bases: torch.nn.modules.module.Module

Image to Patch Embedding

**\_\_init\_\_** (*img\_size*=224, *patch\_size*=16, *in\_chans*=3, *embed\_dim*=768, *norm\_layer*=None)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**flops()**

**training:** bool

```
class easycv.models.backbones.swin_transformer_dynamic.SwinTransformer(img_size=224,
 patch_size=4,
 in_chans=3,
 num_classes=1000,
 embed_dim=96,
 depths=[2, 2, 6, 2],
 num_heads=[3, 6, 12,
 24], window_size=7,
 mlp_ratio=4.0,
 qkv_bias=True,
 qk_scale=None,
 drop_rate=0.0,
 attn_drop_rate=0.0,
 drop_path_rate=0.1,
 norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>,
 ape=False,
 patch_norm=True,
 use_dense_prediction=False,
 **kwargs)
```

Bases: torch.nn.modules.module.Module

**Swin Transformer**

A PyTorch impl of [Swin Transformer: Hierarchical Vision Transformer using Shifted Windows -] <https://arxiv.org/pdf/2103.14030>

**Parameters**

- **img\_size** (*int* | *tuple(int)*) – Input image size.
- **patch\_size** (*int* | *tuple(int)*) – Patch size.
- **in\_chans** (*int*) – Number of input channels.
- **num\_classes** (*int*) – Number of classes for classification head.
- **embed\_dim** (*int*) – Embedding dimension.
- **depths** (*tuple(int)*) – Depth of Swin Transformer layers.
- **num\_heads** (*tuple(int)*) – Number of attention heads in different layers.
- **window\_size** (*int*) – Window size.
- **mlp\_ratio** (*float*) – Ratio of mlp hidden dim to embedding dim.
- **qkv\_bias** (*bool*) – If True, add a learnable bias to query, key, value. Default: True



- **qk\_scale** (*float*) – Override default qk scale of head\_dim \*\* -0.5 if set.
- **drop\_rate** (*float*) – Dropout rate.
- **attn\_drop\_rate** (*float*) – Attention dropout rate.
- **drop\_path\_rate** (*float*) – Stochastic depth rate.
- **norm\_layer** (*nn.Module*) – normalization layer.
- **ape** (*bool*) – If True, add absolute position embedding to the patch embedding.
- **patch\_norm** (*bool*) – If True, add normalization after patch embedding.

```
__init__(img_size=224, patch_size=4, in_chans=3, num_classes=1000, embed_dim=96, depths=[2, 2, 6,
2], num_heads=[3, 6, 12, 24], window_size=7, mlp_ratio=4.0, qkv_bias=True, qk_scale=None,
drop_rate=0.0, attn_drop_rate=0.0, drop_path_rate=0.1, norm_layer=<class
'torch.nn.modules.normalization.LayerNorm'>, ape=False, patch_norm=True,
use_dense_prediction=False, **kwargs)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
init_weights_unused(pretrained=None)
```

```
no_weight_decay()
```

```
no_weight_decay_keywords()
```

```
forward_features(x)
```

```
forward_feature_maps(x)
```

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
forward_selfattention(x, n=1)
```

```
forward_last_selfattention(x)
```

```
forward_all_selfattention(x)
```

```
forward_return_n_last_blocks(x, n=1, return_patch_avgpool=False, depth=[])
```

```
flops()
```

```
init_weights(pretrained="", pretrained_layers=[], verbose=True)
```

```
freeze_pretrained_layers(frozen_layers=[])
```

```
training: bool
```

```
easycv.models.backbones.swin_transformer_dynamic.dynamic_swin_tiny_p4_w7_224(pretrained=False,
**kwargs)
```

```
easycv.models.backbones.swin_transformer_dynamic.dynamic_swin_small_p4_w7_224(pretrained=False,
**kwargs)
```

```
easycv.models.backbones.swin_transformer_dynamic.dynamic_swin_base_p4_w7_224(pretrained=False,
**kwargs)
```

**easycv.models.backbones.vit\_transformer\_dynamic module**

Mostly copy-paste from timm library. [https://github.com/rwightman/pytorch-image-models/blob/master/timm/models/vision\\_transformer.py](https://github.com/rwightman/pytorch-image-models/blob/master/timm/models/vision_transformer.py)

dynamic Input support borrow from [https://github.com/microsoft/esvit/blob/main/models/vision\\_transformer.py](https://github.com/microsoft/esvit/blob/main/models/vision_transformer.py)

`easycv.models.backbones.vit_transformer_dynamic.DropPath(x, drop_prob: float = 0.0, training: bool = False)`

**class** `easycv.models.backbones.vit_transformer_dynamic.DropPath(drop_prob=None)`

Bases: `torch.nn.modules.module.Module`

Drop paths (Stochastic Depth) per sample (when applied in main path of residual blocks).

**\_\_init\_\_**(*drop\_prob=None*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `easycv.models.backbones.vit_transformer_dynamic.Mlp(in_features, hidden_features=None, out_features=None, act_layer=<class 'torch.nn.modules.activation.GELU'>, drop=0.0)`

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**(*in\_features, hidden\_features=None, out\_features=None, act\_layer=<class 'torch.nn.modules.activation.GELU'>, drop=0.0*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `easycv.models.backbones.vit_transformer_dynamic.Attention(dim, num_heads=8, qkv_bias=False, qk_scale=None, attn_drop=0.0, proj_drop=0.0)`

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**(*dim, num\_heads=8, qkv\_bias=False, qk\_scale=None, attn\_drop=0.0, proj\_drop=0.0*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.vit_transformer_dynamic.Block(dim, num_heads, mlp_ratio=4.0,
 qkv_bias=False, qk_scale=None,
 drop=0.0, attn_drop=0.0,
 drop_path=0.0, act_layer=<class
 'torch.nn.modules.activation.GELU'>,
 norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>)
```

Bases: torch.nn.modules.module.Module

```
__init__(dim, num_heads, mlp_ratio=4.0, qkv_bias=False, qk_scale=None, drop=0.0, attn_drop=0.0,
 drop_path=0.0, act_layer=<class 'torch.nn.modules.activation.GELU'>, norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x, return\_attention=False)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**forward\_fea\_and\_attn(x)**

**training:** bool

```
class easycv.models.backbones.vit_transformer_dynamic.PatchEmbed(img_size=224, patch_size=16,
 in_chans=3, embed_dim=768)
```

Bases: torch.nn.modules.module.Module

Image to Patch Embedding

```
__init__(img_size=224, patch_size=16, in_chans=3, embed_dim=768)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.vit_transformer_dynamic.VisionTransformer(img_size=[224],
 patch_size=16,
 in_chans=3,
 num_classes=0,
 embed_dim=768,
 depth=12,
 num_heads=12,
 mlp_ratio=4.0,
 qkv_bias=False,
 qk_scale=None,
 drop_rate=0.0,
 attn_drop_rate=0.0,
 drop_path_rate=0.0,
 norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>,
 use_dense_prediction=False,
 global_pool=False,
 **kwargs)
```

Bases: torch.nn.modules.module.Module

Vision Transformer

```
__init__(img_size=[224], patch_size=16, in_chans=3, num_classes=0, embed_dim=768, depth=12,
 num_heads=12, mlp_ratio=4.0, qkv_bias=False, qk_scale=None, drop_rate=0.0,
 attn_drop_rate=0.0, drop_path_rate=0.0, norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>, use_dense_prediction=False, global_pool=False,
 **kwargs)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(pretrained=None)

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**forward\_features**(x)

**forward\_feature\_maps**(x)

**interpolate\_pos\_encoding**(x, pos\_embed)

**forward\_selfattention**(x, n=1)

**forward\_last\_selfattention**(x)

**forward\_all\_selfattention**(x)

**forward\_return\_n\_last\_blocks**(x, n=1, return\_patch\_avgpool=False, depths=[])

**training:** bool

```
easycv.models.backbones.vit_transformer_dynamic.dynamic_deit_tiny_p16(patch_size=16, **kwargs)
```

```

easycv.models.backbones.vit_transformer_dynamic.dynamic_deit_small_p16(patch_size=16,
 **kwargs)
easycv.models.backbones.vit_transformer_dynamic.dynamic_vit_base_p16(patch_size=16, **kwargs)
easycv.models.backbones.vit_transformer_dynamic.dynamic_vit_large_p16(patch_size=16, **kwargs)
easycv.models.backbones.vit_transformer_dynamic.dynamic_vit_huge_p14(patch_size=14, **kwargs)

```

### easycv.models.backbones.xcit\_transformer module

Implementation of Cross-Covariance Image Transformer (XCiT) Based on timm and DeiT code bases <https://github.com/rwightman/pytorch-image-models/tree/master/timm> <https://github.com/facebookresearch/deit/>

XCiT Transformer. Part of the code is borrowed from: <https://github.com/facebookresearch/xcit/blob/master/xcit.py>

```

class easycv.models.backbones.xcit_transformer.PositionalEncodingFourier(hidden_dim=32,
 dim=768,
 temperature=10000)

```

Bases: torch.nn.modules.module.Module

Positional encoding relying on a fourier kernel matching the one used in the “Attention is all of Need” paper. The implementation builds on DeTR code [https://github.com/facebookresearch/detr/blob/master/models/position\\_encoding.py](https://github.com/facebookresearch/detr/blob/master/models/position_encoding.py)

```
__init__(hidden_dim=32, dim=768, temperature=10000)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(B, H, W)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```

easycv.models.backbones.xcit_transformer.conv3x3(in_planes, out_planes, stride=1)
3x3 convolution with padding

```

```

class easycv.models.backbones.xcit_transformer.ConvPatchEmbed(img_size=224, patch_size=16,
 in_chans=3, embed_dim=768)

```

Bases: torch.nn.modules.module.Module

Image to Patch Embedding using multiple convolutional layers

```
__init__(img_size=224, patch_size=16, in_chans=3, embed_dim=768)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x, padding_size=None)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks

---

while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.xcit_transformer.LPI(in_features, hidden_features=None,
 out_features=None, act_layer=<class
 'torch.nn.modules.activation.GELU'>, drop=0.0,
 kernel_size=3)
```

Bases: torch.nn.modules.module.Module

Local Patch Interaction module that allows explicit communication between tokens in 3x3 windows to augment the implicit communication performed by the block diagonal scatter attention. Implemented using 2 layers of separable 3x3 convolutions with GeLU and BatchNorm2d

```
__init__(in_features, hidden_features=None, out_features=None, act_layer=<class
 'torch.nn.modules.activation.GELU'>, drop=0.0, kernel_size=3)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*, *H*, *W*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.xcit_transformer.ClassAttention(dim, num_heads=8,
 qkv_bias=False, qk_scale=None,
 attn_drop=0.0, proj_drop=0.0)
```

Bases: torch.nn.modules.module.Module

Class Attention Layer as in CaiT <https://arxiv.org/abs/2103.17239>

```
__init__(dim, num_heads=8, qkv_bias=False, qk_scale=None, attn_drop=0.0, proj_drop=0.0)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

---

```
class easycv.models.backbones.xcit_transformer.ClassAttentionBlock(dim, num_heads,
 mlp_ratio=4.0,
 qkv_bias=False,
 qk_scale=None, drop=0.0,
 attn_drop=0.0,
 drop_path=0.0,
 act_layer=<class
 'torch.nn.modules.activation.GELU'>,
 norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>,
 eta=None,
 tokens_norm=False)
```

Bases: torch.nn.modules.module.Module

Class Attention Layer as in CaiT <https://arxiv.org/abs/2103.17239>

```
__init__(dim, num_heads, mlp_ratio=4.0, qkv_bias=False, qk_scale=None, drop=0.0, attn_drop=0.0,
 drop_path=0.0, act_layer=<class 'torch.nn.modules.activation.GELU'>, norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>, eta=None, tokens_norm=False)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(x, H, W, mask=None)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.backbones.xcit_transformer.XCA(dim, num_heads=8, qkv_bias=False,
 qk_scale=None, attn_drop=0.0, proj_drop=0.0)
```

Bases: torch.nn.modules.module.Module

Cross-Covariance Attention (XCA) operation where the channels are updated using a weighted sum.

The weights are obtained from the (softmax normalized) Cross-covariance matrix ( $Q^T K$  in  $d_h$  times  $d_h$ )

```
__init__(dim, num_heads=8, qkv_bias=False, qk_scale=None, attn_drop=0.0, proj_drop=0.0)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**no\_weight\_decay**()

**training:** bool

```
class easycv.models.backbones.xcit_transformer.XCABlock(dim, num_heads, mlp_ratio=4.0,
 qkv_bias=False, qk_scale=None,
 drop=0.0, attn_drop=0.0, drop_path=0.0,
 act_layer=<class
 'torch.nn.modules.activation.GELU'>,
 norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>,
 num_tokens=196, eta=None)
```

Bases: `torch.nn.modules.module.Module`

```
__init__(dim, num_heads, mlp_ratio=4.0, qkv_bias=False, qk_scale=None, drop=0.0, attn_drop=0.0,
 drop_path=0.0, act_layer=<class 'torch.nn.modules.activation.GELU'>, norm_layer=<class
 'torch.nn.modules.normalization.LayerNorm'>, num_tokens=196, eta=None)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*, *H*, *W*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

```
class easycv.models.backbones.xcit_transformer.XCiT(img_size=224, patch_size=16, in_chans=3,
 num_classes=1000, embed_dim=768, depth=12,
 num_heads=12, mlp_ratio=4.0, qkv_bias=True,
 qk_scale=None, drop_rate=0.0,
 attn_drop_rate=0.0, drop_path_rate=0.0,
 norm_layer=None, cls_attn_layers=2,
 use_pos=True, patch_proj='linear', eta=None,
 tokens_norm=False)
```

Bases: `torch.nn.modules.module.Module`

Based on timm and DeiT code bases <https://github.com/rwightman/pytorch-image-models/tree/master/timm>  
<https://github.com/facebookresearch/deit/>

```
__init__(img_size=224, patch_size=16, in_chans=3, num_classes=1000, embed_dim=768, depth=12,
 num_heads=12, mlp_ratio=4.0, qkv_bias=True, qk_scale=None, drop_rate=0.0,
 attn_drop_rate=0.0, drop_path_rate=0.0, norm_layer=None, cls_attn_layers=2, use_pos=True,
 patch_proj='linear', eta=None, tokens_norm=False)
```

#### Parameters

- **img\_size** (*int*, *tuple*) – input image size
- **patch\_size** (*int*, *tuple*) – patch size
- **in\_chans** (*int*) – number of input channels
- **num\_classes** (*int*) – number of classes for classification head
- **embed\_dim** (*int*) – embedding dimension
- **depth** (*int*) – depth of transformer
- **num\_heads** (*int*) – number of attention heads



- **mlp\_ratio** (*int*) – ratio of mlp hidden dim to embedding dim
- **qkv\_bias** (*bool*) – enable bias for qkv if True
- **qk\_scale** (*float*) – override default qk scale of head\_dim \*\* -0.5 if set
- **drop\_rate** (*float*) – dropout rate
- **attn\_drop\_rate** (*float*) – attention dropout rate
- **drop\_path\_rate** (*float*) – stochastic depth rate
- **norm\_layer** – (nn.Module): normalization layer
- **cls\_attn\_layers** – (int) Depth of Class attention layers
- **use\_pos** – (bool) whether to use positional encoding
- **eta** – (float) layerscale initialization value
- **tokens\_norm** – (bool) Whether to normalize all tokens or just the cls\_token in the CA

**init\_weights**(*pretrained=None*)

**no\_weight\_decay**()

**forward\_features**(*x*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** *bool*

`easycv.models.backbones.xcit_transformer.xcit_small_12_p16(pretrained=False, **kwargs)`

`easycv.models.backbones.xcit_transformer.xcit_small_24_p16(pretrained=False, **kwargs)`

`easycv.models.backbones.xcit_transformer.xcit_medium_24_p16(pretrained=False, **kwargs)`

`easycv.models.backbones.xcit_transformer.xcit_small_12_p8(pretrained=False, **kwargs)`

`easycv.models.backbones.xcit_transformer.xcit_small_24_p8(pretrained=False, **kwargs)`

`easycv.models.backbones.xcit_transformer.xcit_medium_24_p8(pretrained=False, **kwargs)`

`easycv.models.backbones.xcit_transformer.xcit_large_24_p8(pretrained=False, **kwargs)`

## 16.1.2 easycv.models.classification package

### Submodules

#### easycv.models.classification.classification module

`easycv.models.classification.classification.distill_loss`(*cls\_score*, *teacher\_score*,  
*tempreature*=1.0)

Soft cross entropy loss

**class** `easycv.models.classification.classification.Classification`(*backbone*, *train\_preprocess*=[],  
*with\_sobel*=False, *head*=None,  
*neck*=None, *teacher*=None,  
*pretrained*=None,  
*mixup\_cfg*=None)

Bases: `easycv.models.base.BaseModel`

**\_\_init\_\_**(*backbone*, *train\_preprocess*=[], *with\_sobel*=False, *head*=None, *neck*=None, *teacher*=None,  
*pretrained*=None, *mixup\_cfg*=None)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(*pretrained*=None)

**forward\_backbone**(*img*: *torch.Tensor*) → List[*torch.Tensor*]

Forward backbone

**Returns** backbone outputs

**Return type** x (tuple)

**forward\_train**(*img*, *gt\_labels*) → Dict[str, *torch.Tensor*]

In forward train, model will forward backbone + neck / multi-neck, get alist of output tensor, and put this list to head / multi-head, to compute each loss

**forward\_test**(*img*: *torch.Tensor*) → Dict[str, *torch.Tensor*]

forward\_test means generate prob/class from image only support one neck + one head

**forward\_test\_label**(*img*, *gt\_labels*) → Dict[str, *torch.Tensor*]

forward\_test\_label means generate prob/class from image only support one neck + one head ps : head init need set the input feature idx

**training**: bool

**aug\_test**(*imgs*)

**forward\_feature**(*img*) → Dict[str, *torch.Tensor*]

**Forward feature means forward backbone + neck/multineck ,get dict of output feature,**

self.neck\_num = 0: means only forward backbone, output backbone feature with avgpool,  
with key neck, self.neck\_num > 0: means has 1/multi neck, output neck's feature with key  
neck\_neckidx\_featureidx, such as neck\_0\_0

**Returns** feature tensor

**Return type** x (*torch.Tensor*)

**update\_extract\_list**(*key*)

**forward**(img: torch.Tensor, mode: str = 'train', gt\_labels: Optional[torch.Tensor] = None, img metas: Optional[torch.Tensor] = None) → Dict[str, torch.Tensor]

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

## easycv.models.classification.necks module

**class** easycv.models.classification.necks.**LinearNeck**(in\_channels, out\_channels, with\_avg\_pool=True, with\_norm=False)

Bases: torch.nn.modules.module.Module

Linear neck: fc only

**\_\_init\_\_**(in\_channels, out\_channels, with\_avg\_pool=True, with\_norm=False)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(init\_linear='normal')

**forward**(x)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**class** easycv.models.classification.necks.**RetrivalNeck**(in\_channels, out\_channels, with\_avg\_pool=True, cdg\_config=['G', 'M'])

Bases: torch.nn.modules.module.Module

**RetrivalNeck:** refer, Combination of Multiple Global Descriptors for Image Retrieval <https://arxiv.org/pdf/1903.10663.pdf>

CGD feature : only use avg pool + gem pooling + max pooling, by pool -> fc -> norm -> concat -> norm Avg feature : use avg pooling, avg pool -> syncbn -> fc

len(cgd\_config) > 0: return [CGD, Avg] len(cgd\_config) = 0 : return [Avg]

**\_\_init\_\_**(in\_channels, out\_channels, with\_avg\_pool=True, cdg\_config=['G', 'M'])

Init RetrivalNeck, faceid neck doesn't pool for input feature map, doesn't support dynamic input

### Parameters

- **in\_channels** – Int - input feature map channels
- **out\_channels** – Int - output feature map channels
- **with\_avg\_pool** – bool do avg pool for BNneck or not
- **cdg\_config** – list('G','M','S'), to configure output feature, CGD = [gempooling] + [maxpooling] + [meanpooling], if len(cgd\_config) > 0: return [CGD, Avg] if len(cgd\_config) = 0 : return [Avg]

**init\_weights**(*init\_linear='normal'*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `easycv.models.classification.necks.FaceIDNeck`(*in\_channels, out\_channels, map\_shape=1, dropout\_ratio=0.4, with\_norm=False, bn\_type='SyncBN'*)

Bases: `torch.nn.modules.module.Module`

FaceID neck: Include BN, dropout, flatten, linear, bn

**\_\_init\_\_**(*in\_channels, out\_channels, map\_shape=1, dropout\_ratio=0.4, with\_norm=False, bn\_type='SyncBN'*)

Init FaceIDNeck, faceid neck doesn't pool for input feature map, doesn't support dynamic input

#### Parameters

- **in\_channels** – Int - input feature map channels
- **out\_channels** – Int - output feature map channels
- **map\_shape** – Int or list(int,...), input feature map (w,h) or w when w=h,
- **dropout\_ratio** – float, drop out ratio
- **with\_norm** – normalize output feature or not
- **bn\_type** – SyncBN or BN

**init\_weights**(*init\_linear='normal'*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `easycv.models.classification.necks.MultiLinearNeck`(*in\_channels, out\_channels, num\_layers=1, with\_avg\_pool=True*)

Bases: `torch.nn.modules.module.Module`

MultiLinearNeck neck: MultiFc head

**\_\_init\_\_**(*in\_channels, out\_channels, num\_layers=1, with\_avg\_pool=True*)

#### Parameters

- **in\_channels** – int or list[int]
- **out\_channels** – int or list[int]
- **num\_layers** – total fc num
- **with\_avg\_pool** – input will be avgPool if True

**Returns** None

**Raises**

- **len(in\_channel) != len(out\_channels)** –
- **len(in\_channel) != len(num\_layers)** –

**init\_weights**(*init\_linear='normal'*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**class** easycv.models.classification.necks.**HRFuseScales**(*in\_channels, out\_channels=2048, norm\_cfg={'momentum': 0.1, 'type': 'BN'}*)

Bases: torch.nn.modules.module.Module

Fuse feature map of multiple scales in HRNet. :param in\_channels: The input channels of all scales. :type in\_channels: list[int] :param out\_channels: The channels of fused feature map.

Defaults to 2048.

**Parameters**

- **norm\_cfg**(*dict*) – dictionary to construct norm layers. Defaults to dict(type='BN', momentum=0.1).
- **init\_cfg**(*dict | list[dict], optional*) – Initialization config dict. Defaults to dict(type='Normal', layer='Linear', std=0.01).

**\_\_init\_\_**(*in\_channels, out\_channels=2048, norm\_cfg={'momentum': 0.1, 'type': 'BN'}*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**training:** bool

**init\_weights**(*init\_linear='normal'*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

### 16.1.3 easycv.models.detection package

#### Subpackages

`easycv.models.detection.utils` package

#### Submodules

`easycv.models.detection.utils.bboxes` module

```
easycv.models.detection.utils.bboxes.bboxes_iou(bboxes_a, bboxes_b, xyxy=True)
easycv.models.detection.utils.bboxes.postprocess(prediction, num_classes, conf_thre=0.7,
 nms_thre=0.45)
```

`easycv.models.detection.yolox` package

#### Submodules

`easycv.models.detection.yolox.yolo_head` module

```
class easycv.models.detection.yolox.yolo_head.YOLOXHead(num_classes, width=1.0, strides=[8, 16,
 32], in_channels=[256, 512, 1024],
 act='silu', depthwise=False,
 stage='CLOUD')
```

Bases: `torch.nn.modules.module.Module`

```
__init__(num_classes, width=1.0, strides=[8, 16, 32], in_channels=[256, 512, 1024], act='silu',
 depthwise=False, stage='CLOUD')
```

#### Parameters

- **num\_classes** (*int*) – detection class numbers.
- **width** (*float*) – model width. Default value: 1.0.
- **strides** (*list*) – expanded strides. Default value: [8, 16, 32].
- **in\_channels** (*list*) – model conv channels set. Default value: [256, 512, 1024].
- **act** (*str*) – activation type of conv. Defalut value: “silu”.
- **depthwise** (*bool*) – whether apply depthwise conv in conv branch. Default value: False.
- **stage** (*str*) – model stage, distinguish edge head to cloud head. Default value: CLOUD.

**initialize\_biases**(*prior\_prob*)

**forward**(*xin*, *labels=None*, *imgs=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```

get_output_and_grid(output, k, stride, dtype)
decode_outputs(outputs, dtype)
get_losses(imgs, x_shifts, y_shifts, expanded_strides, labels, outputs, origin_preds, dtype)
get_l1_target(l1_target, gt, stride, x_shifts, y_shifts, eps=1e-08)
get_assignments(batch_idx, num_gt, total_num_anchors, gt_bboxes_per_image, gt_classes,
 bboxes_preds_per_image, expanded_strides, x_shifts, y_shifts, cls_preds, bbox_preds,
 obj_preds, labels, imgs, mode='gpu')
get_in_boxes_info(gt_bboxes_per_image, expanded_strides, x_shifts, y_shifts, total_num_anchors,
 num_gt)
dynamic_k_matching(cost, pair_wise_iious, gt_classes, num_gt, fg_mask)
training: bool

```

### easy cv.models.detection.yolox.yolo\_pafpn module

```

class easy cv.models.detection.yolox.yolo_pafpn.YOLOPAFPN(depth=1.0, width=1.0,
 in_features=('dark3', 'dark4', 'dark5'),
 in_channels=[256, 512, 1024],
 depthwise=False, act='silu')

```

Bases: torch.nn.modules.module.Module

YOLOv3 model. Darknet 53 is the default backbone of this model.

```

__init__(depth=1.0, width=1.0, in_features=('dark3', 'dark4', 'dark5'), in_channels=[256, 512, 1024],
 depthwise=False, act='silu')

```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```

forward(input)

```

**Parameters** **inputs** – input images.

**Returns** FPN feature.

**Return type** Tuple[Tensor]

```

training: bool

```

**easycv.models.detection.yolox.yolox module****easycv.models.detection.yolox.yolox.init\_yolo**(*M*)**class** easycv.models.detection.yolox.yolox.**YOLOX**(*model\_type: str = 's', num\_classes: int = 80, test\_size: tuple = (640, 640), test\_conf: float = 0.01, nms\_thre: float = 0.65, pretrained: Optional[str] = None*)Bases: [easycv.models.base.BaseModel](#)

YOLOX model module. The module list is defined by create\_yolov3\_modules function. The network returns loss values from three YOLO layers during training and detection results during test.

**param\_map** = {'l': [1.0, 1.0], 'm': [0.67, 0.75], 'nano': [0.33, 0.25], 's': [0.33, 0.5], 'tiny': [0.33, 0.375], 'x': [1.33, 1.25]}**\_\_init\_\_**(*model\_type: str = 's', num\_classes: int = 80, test\_size: tuple = (640, 640), test\_conf: float = 0.01, nms\_thre: float = 0.65, pretrained: Optional[str] = None*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward\_train**(*img: torch.Tensor, gt\_bboxes: torch.Tensor, gt\_labels: torch.Tensor, img metas=None, scale=None*) → Dict[str, torch.Tensor]

Abstract interface for model forward in training

**Parameters**

- **img** (*Tensor*) – image tensor, NxCxHxW
- **target** (*List[Tensor]*) – list of target tensor, NTx5 [class,x\_c,y\_c,w,h]

**forward\_test**(*img: torch.Tensor, img\_metas=None*) → torch.Tensor

Abstract interface for model forward in training

**Parameters**

- **img** (*Tensor*) – image tensor, NxCxHxW
- **target** (*List[Tensor]*) – list of target tensor, NTx5 [class,x\_c,y\_c,w,h]

**forward**(*img, mode='compression', \*\*kwargs*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**forward\_compression**(*x*)**training:** bool



## easycv.models.detection.yolox\_edge package

### Submodules

#### easycv.models.detection.yolox\_edge.yolox\_edge module

easycv.models.detection.yolox\_edge.yolox\_edge.**init\_yolo**(M)

```
class easycv.models.detection.yolox_edge.yolox_edge.YOLOX_EDGE(stage: str = 'EDGE', model_type:
 str = 's', num_classes: int = 80,
 test_size: tuple = (640, 640),
 test_conf: float = 0.01, nms_thre:
 float = 0.65, pretrained:
 Optional[str] = None, depth: float =
 1.0, width: float = 1.0,
 max_model_params: float = 0.0,
 max_model_flops: float = 0.0,
 activation: str = 'silu',
 in_channels: list = [256, 512,
 1024], backbone=None,
 head=None)
```

Bases: [easycv.models.detection.yolox.yolox.YOLOX](#)

YOLOX model module. The module list is defined by create\_yolov3\_modules function. The network returns loss values from three YOLO layers during training and detection results during test.

```
__init__(stage: str = 'EDGE', model_type: str = 's', num_classes: int = 80, test_size: tuple = (640, 640),
 test_conf: float = 0.01, nms_thre: float = 0.65, pretrained: Optional[str] = None, depth: float =
 1.0, width: float = 1.0, max_model_params: float = 0.0, max_model_flops: float = 0.0, activation:
 str = 'silu', in_channels: list = [256, 512, 1024], backbone=None, head=None)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**training:** bool

## 16.1.4 easycv.models.heads package

### Submodules

#### easycv.models.heads.cls\_head module

```
class easycv.models.heads.cls_head.ClsHead(with_avg_pool=False, label_smooth=0.0,
 in_channels=2048, with_fc=True, num_classes=1000,
 loss_config={'type': 'CrossEntropyLossWithLabelSmooth'},
 input_feature_index=[0])
```

Bases: [torch.nn.modules.module.Module](#)

Simplest classifier head, with only one fc layer. Should Notice Evtorch module design input always be feature\_list = [tensor, tensor,...]

```
__init__(with_avg_pool=False, label_smooth=0.0, in_channels=2048, with_fc=True, num_classes=1000,
 loss_config={'type': 'CrossEntropyLossWithLabelSmooth'}, input_feature_index=[0])
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(pretrained=None, init\_linear='normal', std=0.01, bias=0.0)

**forward**(*x*: *List[torch.Tensor]*) → *List[torch.Tensor]*

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**loss**(*cls\_score*: *List[torch.Tensor]*, *labels*: *torch.Tensor*) → *Dict[str, torch.Tensor]*

#### Parameters

- **cls\_score** – [N x num\_classes]
- **labels** – if don't use mixup, shape is [N], else [N x num\_classes]

**mixup\_loss**(*cls\_score*, *labels\_1*, *labels\_2*, *lam*) → *Dict[str, torch.Tensor]*

**training:** *bool*

### easycv.models.heads.contrastive\_head module

**class** easycv.models.heads.contrastive\_head.**ContrastiveHead**(*temperature=0.1*)

Bases: *torch.nn.modules.module.Module*

Head for contrastive learning.

**\_\_init\_\_**(*temperature=0.1*)

Initializes internal Module state, shared by both *nn.Module* and *ScriptModule*.

**forward**(*pos*, *neg*)

#### Parameters

- **pos** (*Tensor*) – Nx1 positive similarity
- **neg** (*Tensor*) – Nxk negative similarity

**training:** *bool*

**class** easycv.models.heads.contrastive\_head.**DebiasedContrastiveHead**(*temperature=0.1*, *tau=0.1*)

Bases: *torch.nn.modules.module.Module*

**\_\_init\_\_**(*temperature=0.1*, *tau=0.1*)

Initializes internal Module state, shared by both *nn.Module* and *ScriptModule*.

**forward**(*pos*, *neg*)

#### Parameters

- **pos** (*Tensor*) – Nx1 positive similarity
- **neg** (*Tensor*) – Nxk negative similarity

**training:** *bool*

**easycv.models.heads.latent\_pred\_head module**

**class** easycv.models.heads.latent\_pred\_head.**LatentPredictHead**(*predictor, size\_average=True*)  
 Bases: torch.nn.modules.module.Module

Head for contrastive learning.

**\_\_init\_\_**(*predictor, size\_average=True*)  
 Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(*init\_linear='normal'*)

**forward**(*input, target*)

**Parameters**

- **input** (*Tensor*) – NxC input features.
- **target** (*Tensor*) – NxC target features.

**training:** bool

**class** easycv.models.heads.latent\_pred\_head.**LatentClsHead**(*predictor*)  
 Bases: torch.nn.modules.module.Module

Head for contrastive learning.

**\_\_init\_\_**(*predictor*)  
 Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(*init\_linear='normal'*)

**forward**(*input, target*)

**Parameters**

- **input** (*Tensor*) – NxC input features.
- **target** (*Tensor*) – NxC target features.

**training:** bool

**easycv.models.heads.mp\_metric\_head module**

easycv.models.heads.mp\_metric\_head.**EmbeddingExpansion**(*embs, labels, explansion\_rate=4, alpha=1.0*)  
 Expand embedding: CVPR refer to <https://github.com/clovaai/embedding-expansion> combine PK sampled data, mixup anchor positive pair to generate more features, always combine with BatchHardminer. result on SOP and CUB need to be add

**Parameters**

- **embs** – [N , dims] tensor
- **labels** – [N] tensor
- **explansion\_rate** – to expand N to explansion\_rate \* N
- **alpha** – beta distribution parameter for mixup

**Returns** [N\*explansion\_rate , dims]

**Return type** embs

```
class easycv.models.heads.mp_metric_head.MpMetricHead(with_avg_pool=False, in_channels=2048,
 loss_config=[{'type': 'CircleLoss',
 'loss_weight': 1.0, 'norm': True, 'ddp': True,
 'm': 0.4, 'gamma': 80}],
 input_feature_index=0,
 input_label_index=0, ignore_label=None)
```

Bases: torch.nn.modules.module.Module

Simplest classifier head, with only one fc layer.

```
__init__(with_avg_pool=False, in_channels=2048, loss_config=[{'type': 'CircleLoss', 'loss_weight': 1.0,
 'norm': True, 'ddp': True, 'm': 0.4, 'gamma': 80}], input_feature_index=0, input_label_index=0,
 ignore_label=None)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
init_weights(pretrained=None, init_linear='normal', std=0.01, bias=0.0)
```

```
forward(x: List[torch.Tensor]) → List[torch.Tensor]
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
loss(cls_score, labels) → Dict[str, torch.Tensor]
```

```
training: bool
```

## easycv.models.heads.multi\_cls\_head module

```
class easycv.models.heads.multi_cls_head.MultiClsHead(pool_type='adaptive', in_indices=(0),
 with_last_layer_unpool=False,
 backbone='resnet50', norm_cfg={'type':
 'BN'}, num_classes=1000)
```

Bases: torch.nn.modules.module.Module

Multiple classifier heads.

```
FEAT_CHANNELS = {'resnet50': [64, 256, 512, 1024, 2048]}
```

```
FEAT_LAST_UNPOOL = {'resnet50': 100352}
```

```
__init__(pool_type='adaptive', in_indices=(0), with_last_layer_unpool=False, backbone='resnet50',
 norm_cfg={'type': 'BN'}, num_classes=1000)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
init_weights()
```

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks

---

while the latter silently ignores them.

```
loss(cls_score, labels)
```

```
training: bool
```

## 16.1.5 easycv.models.loss package

### Submodules

#### easycv.models.loss.iou\_loss module

```
class easycv.models.loss.iou_loss.IOULoss(reduction='none', loss_type='iou')
```

Bases: torch.nn.modules.module.Module

```
__init__(reduction='none', loss_type='iou')
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(pred, target)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
training: bool
```

#### easycv.models.loss.mse\_loss module

```
class easycv.models.loss.mse_loss.JointsMSELoss(use_target_weight=False, loss_weight=1.0)
```

Bases: torch.nn.modules.module.Module

MSE loss for heatmaps.

#### Parameters

- **use\_target\_weight** (*bool*) – Option to use weighted MSE loss. Different joint types may have different target weights.
- **loss\_weight** (*float*) – Weight of the loss. Default: 1.0.

```
__init__(use_target_weight=False, loss_weight=1.0)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(output, target, target_weight)
```

Forward function.

```
training: bool
```

**easycv.models.loss.pytorch\_metric\_learning module**

```
class easycv.models.loss.pytorch_metric_learning.FocalLoss2d(gamma=2, weight=None,
 size_average=None, reduce=None,
 reduction='mean', num_classes=2)
```

Bases: torch.nn.modules.loss.\_WeightedLoss

```
__init__(gamma=2, weight=None, size_average=None, reduce=None, reduction='mean', num_classes=2)
 FocalLoss2d, loss solve 2-class classification unbalance problem
```

**Parameters**

- **gamma** – focal loss param Gamma
- **weight** – weight same as loss.\_WeightedLoss
- **size\_average** – size\_average same as loss.\_WeightedLoss
- **reduce** – reduce same as loss.\_WeightedLoss
- **reduction** – reduce same as loss.\_WeightedLoss
- **num\_classes** – fix num 2

**Returns** Focalloss nn.module.loss object

```
forward(input, target)
 input: [N * num_classes] target : [N * num_classes] one-hot
```

**reduction:** str

```
class easycv.models.loss.pytorch_metric_learning.DistributeMSELoss
```

Bases: torch.nn.modules.module.Module

```
__init__()
 DistributeMSELoss : for faceid age, score predict (regression by softmax)
```

```
forward(input, target)
 Defines the computation performed at every call.
```

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.loss.pytorch_metric_learning.CrossEntropyLossWithLabelSmooth(label_smooth=0.1,
 tempera-
 ture=1.0,
 with_cls=False,
 embed-
 ding_size=512,
 num_classes=10000)
```

Bases: torch.nn.modules.module.Module

```
__init__(label_smooth=0.1, temperature=1.0, with_cls=False, embedding_size=512, num_classes=10000)
 A softmax loss , with label_smooth and fc(to fit pytorch metric learning interface) :param label_smooth:
 label_smooth args, default=0.1 :param with_cls: if True, will generate a nn.Linear to trans input embedding
 from embedding_size to num_classes :param embedding_size: if input is feature not logits, then need this
```

to indicate embedding shape :param num\_classes: if input is feature not logits, then need this to indicate classification num\_classes

**Returns** None

**Raises** **IOError** – An error occurred accessing the bigtable.Table object.

**forward**(*input, target*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.loss.pytorch_metric_learning.AMSoftmaxLoss(embedding_size=512,
 num_classes=100000,
 margin=0.35, scale=30)
```

Bases: torch.nn.modules.module.Module

**\_\_init\_\_**(*embedding\_size=512, num\_classes=100000, margin=0.35, scale=30*)

AMsoftmax loss , with fc(to fit pytorch metric learning interface), paper: <https://arxiv.org/pdf/1801.05599.pdf> :param embedding\_size: forward input [N, embedding\_size ] :param num\_classes: classification num\_classes :param margin: AMSoftmax param :param scale: AMSoftmax param, should increase num\_classes

**forward**(*x, lb*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.loss.pytorch_metric_learning.ModelParallelSoftmaxLoss(embedding_size=512,
 num_classes=100000,
 scale=None,
 margin=None,
 bias=True)
```

Bases: torch.nn.modules.module.Module

**\_\_init\_\_**(*embedding\_size=512, num\_classes=100000, scale=None, margin=None, bias=True*)

ModelParallel Softmax by sailfish :param embedding\_size: forward input [N, embedding\_size ] :param num\_classes: classification num\_classes

**forward**(*x, lb*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the

---

Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.loss.pytorch_metric_learning.ModelParallelAMSoftmaxLoss(embedding_size=512,
 num_classes=100000,
 margin=0.35,
 scale=30)
```

Bases: torch.nn.modules.module.Module

**\_\_init\_\_**(embedding\_size=512, num\_classes=100000, margin=0.35, scale=30)

ModelParallel AMSOftmax by sailfish :param embedding\_size: forward input [N, embedding\_size ]  
:param num\_classes: classification num\_classes

**forward**(x, lb)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.loss.pytorch_metric_learning.SoftTargetCrossEntropy(num_classes=1000,
 **kwargs)
```

Bases: torch.nn.modules.module.Module

**\_\_init\_\_**(num\_classes=1000, \*\*kwargs)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(x: torch.Tensor, target: torch.Tensor) → torch.Tensor

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

## 16.1.6 easycv.models.pose package

### Subpackages

#### easycv.models.pose.heads package

### Submodules



**easycv.models.pose.heads.topdown\_heatmap\_base\_head module**

**class** easycv.models.pose.heads.topdown\_heatmap\_base\_head.TopdownHeatmapBaseHead

Bases: torch.nn.modules.module.Module

Base class for top-down heatmap heads.

All top-down heatmap heads should subclass it. All subclass should overwrite:

Methods:*get\_loss*, supporting to calculate loss. Methods:*get\_accuracy*, supporting to calculate accuracy. Methods:*forward*, supporting to forward model. Methods:*inference\_model*, supporting to inference model.

**abstract** *get\_loss*(\*\*kwargs)

Gets the loss.

**abstract** *get\_accuracy*(\*\*kwargs)

Gets the accuracy.

**abstract** *forward*(\*\*kwargs)

Forward function.

**abstract** *inference\_model*(\*\*kwargs)

Inference function.

**decode**(img metas, output, \*\*kwargs)

Decode keypoints from heatmaps.

**Parameters**

- **img\_metas** (*list(dict)*) – Information about data augmentation By default this includes: - “image\_file”: path to the image file - “center”: center of the bbox - “scale”: scale of the bbox - “rotation”: rotation of the bbox - “bbox\_score”: score of bbox
- **output** (*np.ndarray[N, K, H, W]*) – model predicted heatmaps.

**training:** bool

**easycv.models.pose.heads.topdown\_heatmap\_simple\_head module**

**class** easycv.models.pose.heads.topdown\_heatmap\_simple\_head.TopdownHeatmapSimpleHead(*in\_channels*,  
*out\_channels*,  
*num\_deconv\_layers*=3,  
*num\_deconv\_filters*=(256,  
256,  
256),  
*num\_deconv\_kernels*=(4,  
4, 4),  
*ex-  
tra*=None,  
*in\_index*=0,  
*in-  
put\_transform*=None,  
*align\_corners*=False,  
*loss\_keypoint*=None,  
*train\_cfg*=None,  
*test\_cfg*=None)

Bases: easycv.models.pose.heads.topdown\_heatmap\_base\_head.TopdownHeatmapBaseHead

Top-down heatmap simple head. paper ref: Bin Xiao et al. Simple Baselines for Human Pose Estimation and Tracking.

TopdownHeatmapSimpleHead is consisted of ( $\geq 0$ ) number of deconv layers and a simple conv2d layer.

#### Parameters

- **in\_channels** (*int*) – Number of input channels
- **out\_channels** (*int*) – Number of output channels
- **num\_deconv\_layers** (*int*) – Number of deconv layers. num\_deconv\_layers should  $\geq 0$ . Note that 0 means no deconv layers.
- **num\_deconv\_filters** (*list/tuple*) – Number of filters. If num\_deconv\_layers  $> 0$ , the length of
- **num\_deconv\_kernels** (*list/tuple*) – Kernel sizes.
- **in\_index** (*int/Sequence[int]*) – Input feature index. Default: 0
- **input\_transform** (*str/None*) – Transformation type of input features. Options: 'resize\_concat', 'multiple\_select', None. Default: None.
  - **'resize\_concat'**: Multiple feature maps will be resized to the same size as the first one and then concat together. Usually used in FCN head of HRNet.
  - **'multiple\_select'**: Multiple feature maps will be bundle into a list and passed into decode head.
  - None: Only one select feature map is allowed.
- **align\_corners** (*bool*) – align\_corners argument of F.interpolate. Default: False.
- **loss\_keypoint** (*dict*) – Config for keypoint loss. Default: None.

**\_\_init\_\_** (*in\_channels, out\_channels, num\_deconv\_layers=3, num\_deconv\_filters=(256, 256, 256), num\_deconv\_kernels=(4, 4, 4), extra=None, in\_index=0, input\_transform=None, align\_corners=False, loss\_keypoint=None, train\_cfg=None, test\_cfg=None*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**get\_loss** (*output, target, target\_weight*)  
Calculate top-down keypoint loss.

---

**Note:** batch\_size: N num\_keypoints: K heatmaps height: H heatmaps weight: W

---

#### Parameters

- **output** (*torch.Tensor[NxKxHxW]*) – Output heatmaps.
- **target** (*torch.Tensor[NxKxHxW]*) – Target heatmaps.
- **target\_weight** (*torch.Tensor[NxKx1]*) – Weights across different joint types.

**get\_accuracy** (*output, target, target\_weight*)  
Calculate accuracy for top-down keypoint loss.

---

**Note:** batch\_size: N num\_keypoints: K heatmaps height: H heatmaps weight: W

---

#### Parameters

- **output** (*torch.Tensor[NxKxHxW]*) – Output heatmaps.

- **target** (*torch.Tensor*[*NxKxHxW*]) – Target heatmaps.
- **target\_weight** (*torch.Tensor*[*NxKx1*]) – Weights across different joint types.

**forward**(*x*)

Forward function.

**inference\_model**(*x*, *flip\_pairs=None*)

Inference function.

**Returns** Output heatmaps.

**Return type** output\_heatmap (np.ndarray)

**Parameters**

- **x** (*torch.Tensor*[*NxKxHxW*]) – Input features.
- **flip\_pairs** (*None* | *list[tuple()]*) – Pairs of keypoints which are mirrored.

**training:** bool

**init\_weights**()

Initialize model weights.

## Submodules

### easycv.models.pose.top\_down module

**class** easycv.models.pose.top\_down.**TopDown**(*backbone*, *neck=None*, *keypoint\_head=None*,  
*train\_cfg=None*, *test\_cfg=None*, *pretrained=None*,  
*loss\_pose=None*)

Bases: [easycv.models.base.BaseModel](#)

Top-down pose detectors.

**Parameters**

- **backbone** (*dict*) – Backbone modules to extract feature.
- **keypoint\_head** (*dict*) – Keypoint head to process feature.
- **train\_cfg** (*dict*) – Config for training. Default: None.
- **test\_cfg** (*dict*) – Config for testing. Default: None.
- **pretrained** (*str*) – Path to the pretrained models.
- **loss\_pose** (*None*) – Deprecated arguments. Please use *loss\_keypoint* for heads instead.

**\_\_init\_\_**(*backbone*, *neck=None*, *keypoint\_head=None*, *train\_cfg=None*, *test\_cfg=None*, *pretrained=None*,  
*loss\_pose=None*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**property with\_neck**

Check if has keypoint\_head.

**property with\_keypoint**

Check if has keypoint\_head.

**init\_weights**(*pretrained=None*)

Weight initialization for model.

**forward\_train**(*img*, *target*, *target\_weight*, *img metas*, *\*\*kwargs*)

Defines the computation performed at every call when training.

**forward\_test**(*img*, *img metas*, *return\_heatmap=False*, *\*\*kwargs*)

Defines the computation performed at every call when testing.

**show\_result**(*img*, *result*, *skeleton=None*, *kpt\_score\_thr=0.3*, *bbox\_color='green'*, *pose\_kpt\_color=None*, *pose\_link\_color=None*, *text\_color='white'*, *radius=4*, *thickness=1*, *font\_scale=0.5*, *bbox\_thickness=1*, *win\_name=""*, *show=False*, *show\_keypoint\_weight=False*, *wait\_time=0*, *out\_file=None*)

Draw *result* over *img*.

#### Parameters

- **img** (*str* or *Tensor*) – The image to be displayed.
- **result** (*list[dict]*) – The results to draw over *img* (bbox\_result, pose\_result).
- **skeleton** (*list[list]*) – The connection of keypoints. skeleton is 0-based indexing.
- **kpt\_score\_thr** (*float*, *optional*) – Minimum score of keypoints to be shown. Default: 0.3.
- **bbox\_color** (*str* or *tuple* or *Color*) – Color of bbox lines.
- **pose\_kpt\_color** (*np.array[Nx3]*) – Color of N keypoints. If None, do not draw keypoints.
- **pose\_link\_color** (*np.array[Mx3]*) – Color of M links. If None, do not draw links.
- **text\_color** (*str* or *tuple* or *Color*) – Color of texts.
- **radius** (*int*) – Radius of circles.
- **thickness** (*int*) – Thickness of lines.
- **font\_scale** (*float*) – Font scales of texts.
- **win\_name** (*str*) – The window name.
- **show** (*bool*) – Whether to show the image. Default: False.
- **show\_keypoint\_weight** (*bool*) – Whether to change the transparency using the predicted confidence scores of keypoints.
- **wait\_time** (*int*) – Value of waitKey param. Default: 0.
- **out\_file** (*str* or *None*) – The filename to write the image. Default: None.

**Returns** Visualized img, only if not *show* or *out\_file*.

**Return type** Tensor

**training:** bool

## 16.1.7 easycv.models.selfsup package

### Submodules

#### easycv.models.selfsup.byol module

**class** easycv.models.selfsup.byol.**BYOL**(*backbone*, *neck=None*, *head=None*, *pretrained=None*, *base\_momentum=0.996*, *\*\*kwargs*)

Bases: [easycv.models.base.BaseModel](#)

BYOL unofficial implementation. Paper: <https://arxiv.org/abs/2006.07733>

**\_\_init\_\_**(*backbone*, *neck=None*, *head=None*, *pretrained=None*, *base\_momentum=0.996*, *\*\*kwargs*)  
Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(*pretrained=None*)

**forward\_train**(*img*, *\*\*kwargs*)  
Abstract interface for model forward in training

#### Parameters

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward\_test**(*img*, *\*\*kwargs*)  
Abstract interface for model forward in testing

#### Parameters

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward**(*img*, *mode='train'*, *\*\*kwargs*)  
Defines the computation performed at every call.  
Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

#### easycv.models.selfsup.dino module

**class** easycv.models.selfsup.dino.**MultiCropWrapper**(*backbone*, *head*)

Bases: `torch.nn.modules.module.Module`

Perform forward pass separately on each resolution input. The inputs corresponding to a single resolution are clubbed and single forward is run on the same resolution inputs. Hence we do several forward passes = number of different resolutions used. We then concatenate all the output features and run the head forward on these concatenated features.

**\_\_init\_\_**(*backbone*, *head*)  
Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training: bool**

```
class easycv.models.selfsup.dino.DINOLoss(out_dim, ncrops, warmup_teacher_temp, teacher_temp,
 warmup_teacher_temp_epochs, nepochs, device,
 student_temp=0.1, center_momentum=0.9)
```

Bases: `torch.nn.modules.module.Module`

```
__init__(out_dim, ncrops, warmup_teacher_temp, teacher_temp, warmup_teacher_temp_epochs, nepochs,
 device, student_temp=0.1, center_momentum=0.9)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(student_output, teacher_output, epoch)
```

Cross-entropy between softmax outputs of the teacher and student networks.

```
update_center(teacher_output)
```

Update center used for teacher output.

**training: bool**

```
easycv.models.selfsup.dino.has_batchnorms(model)
```

```
easycv.models.selfsup.dino.get_params_groups(model)
```

```
class easycv.models.selfsup.dino.DINO(backbone, train_preprocess=[], neck=None, config=None,
 pretrained=None)
```

Bases: `easycv.models.base.BaseModel`

```
__init__(backbone, train_preprocess=[], neck=None, config=None, pretrained=None)
```

Init Moby

**Parameters**

- **backbone** – backbone config to build vision backbone
- **train\_preprocess** – [gaussBlur, mixUp, solarize]
- **neck** – neck config to build Moby Neck
- **config** – DINO parameter config

```
get_params_groups()
```

```
init_weights(pretrained=None)
```

```
init_before_train()
```

```
momentum_update_key_encoder(m=0.999)
```

ema for dino

```
forward_train(inputs)
```

Abstract interface for model forward in training

**Parameters**

- **img** (*Tensor*) – image tensor

- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward\_test**(*img*, **\*\*kwargs**)

Abstract interface for model forward in testing

**Parameters**

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward\_feature**(*img*, **\*\*kwargs**)

Forward backbone

**Returns** feature tensor

**Return type** x (*torch.Tensor*)

**training:** bool

**forward**(*img*, *gt\_label=None*, *mode='train'*, *extract\_list=['neck']*, **\*\*kwargs**)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

## easycv.models.selfsup.mae module

**class** easycv.models.selfsup.mae.**MAE**(*backbone*, *neck*, *mask\_ratio=0.75*, *norm\_pix\_loss=True*, **\*\*kwargs**)

Bases: [easycv.models.base.BaseModel](#)

**\_\_init\_\_**(*backbone*, *neck*, *mask\_ratio=0.75*, *norm\_pix\_loss=True*, **\*\*kwargs**)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**patchify**(*imgs*)

convert image to patch

**Parameters** **imgs** – (N, 3, H, W)

**Returns** (N, L, patch\_size\*\*2 \*3)

**Return type** x

**forward\_loss**(*imgs*, *pred*, *mask*)

compute loss

**Parameters**

- **imgs** – (N, 3, H, W)
- **pred** – (N, L, p\*p\*3)
- **mask** – (N, L), 0 is keep, 1 is remove,

**forward\_train**(*img*, **\*\*kwargs**)

Abstract interface for model forward in training

**Parameters**

- **img** (*Tensor*) – image tensor

- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward\_test**(*img*, **\*\*kwargs**)

Abstract interface for model forward in testing

#### Parameters

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward**(*img*, *mode*='train', **\*\*kwargs**)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

### easycv.models.selfsup.mixco module

**class** easycv.models.selfsup.mixco.**MIXCO**(*backbone*, *train\_preprocess*=[], *neck*=None, *head*=None, *mixco\_head*=None, *pretrained*=None, *queue\_len*=65536, *feat\_dim*=128, *momentum*=0.999, **\*\*kwargs**)

Bases: [easycv.models.selfsup.moco.MOCO](#)

MOCO.

A mixup version moco <https://arxiv.org/pdf/2010.06300.pdf>

**\_\_init\_\_**(*backbone*, *train\_preprocess*=[], *neck*=None, *head*=None, *mixco\_head*=None, *pretrained*=None, *queue\_len*=65536, *feat\_dim*=128, *momentum*=0.999, **\*\*kwargs**)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward\_train**(*img*, **\*\*kwargs**)

Abstract interface for model forward in training

#### Parameters

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**training:** bool

### easycv.models.selfsup.moby module

**class** easycv.models.selfsup.moby.**MoBY**(*backbone*, *train\_preprocess*=[], *neck*=None, *head*=None, *pretrained*=None, *queue\_len*=4096, *contrast\_temperature*=0.2, *momentum*=0.99, *online\_drop\_path\_rate*=0.2, *target\_drop\_path\_rate*=0.0, **\*\*kwargs**)

Bases: [easycv.models.base.BaseModel](#)

MoBY. Part of the code is borrowed from: <https://github.com/SwinTransformer/Transformer-SSL/blob/main/models/moby.py>.



```
__init__(backbone, train_preprocess=[], neck=None, head=None, pretrained=None, queue_len=4096,
 contrast_temperature=0.2, momentum=0.99, online_drop_path_rate=0.2,
 target_drop_path_rate=0.0, **kwargs)
```

Init Moby

#### Parameters

- **backbone** – backbone config to build vision backbone
- **train\_preprocess** – [gaussBlur, mixUp, solarize]
- **neck** – neck config to build Moby Neck
- **head** – head config to build Moby Neck
- **pretrained** – pretrained weight for backbone
- **queue\_len** – moby queue length
- **contrast\_temperature** – contrastive\_loss temperature
- **momentum** – ema target weights momentum
- **online\_drop\_path\_rate** – for transformer based backbone, set online model drop\_path\_rate
- **target\_drop\_path\_rate** – for transformer based backbone, set target model drop\_path\_rate

```
init_weights(pretrained=None)
```

```
forward_backbone(img)
```

```
contrastive_loss(q, k, queue)
```

```
forward_train(img, **kwargs)
```

Abstract interface for model forward in training

#### Parameters

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

```
forward_test(img, **kwargs)
```

Abstract interface for model forward in testing

#### Parameters

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

```
forward_feature(img, **kwargs)
```

Forward backbone

**Returns** feature tensor

**Return type** x (*torch.Tensor*)

```
forward(img, gt_label=None, mode='train', extract_list=['neck'], **kwargs)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

`easycv.models.selfsup.moby.concat_all_gather(tensor)`

Performs all\_gather operation on the provided tensors. \* **Warning** \*: torch.distributed.all\_gather has no gradient.

### **easycv.models.selfsup.moco module**

**class** `easycv.models.selfsup.moco.MOCO(backbone, train_preprocess=[], neck=None, head=None, pretrained=None, queue_len=65536, feat_dim=128, momentum=0.999, **kwargs)`

Bases: `easycv.models.base.BaseModel`

MOCO. Part of the code is borrowed from: <https://github.com/facebookresearch/moco/blob/master/moco/builder.py>.

**\_\_init\_\_**(`backbone, train_preprocess=[], neck=None, head=None, pretrained=None, queue_len=65536, feat_dim=128, momentum=0.999, **kwargs`)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(`pretrained=None`)

**forward\_backbone**(`img`)

**forward\_train**(`img, **kwargs`)

Abstract interface for model forward in training

#### **Parameters**

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward\_test**(`img, **kwargs`)

Abstract interface for model forward in testing

#### **Parameters**

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward\_feature**(`img, **kwargs`)

Forward backbone

**Returns** feature tensor

**Return type** x (*torch.Tensor*)

**forward**(`img, gt_label=None, mode='train', extract_list=['neck'], **kwargs`)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks

---

while the latter silently ignores them.

**training:** bool

`easycv.models.selfsup.moco.concat_all_gather(tensor)`

Performs all\_gather operation on the provided tensors. \* **Warning** \*: torch.distributed.all\_gather has no gradient.

### easycv.models.selfsup.necks module

**class** `easycv.models.selfsup.necks.DINONeck`(*in\_dim, out\_dim, use\_bn=False, norm\_last\_layer=True, nlayers=3, hidden\_dim=2048, bottleneck\_dim=256*)

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**(*in\_dim, out\_dim, use\_bn=False, norm\_last\_layer=True, nlayers=3, hidden\_dim=2048, bottleneck\_dim=256*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**class** `easycv.models.selfsup.necks.MoBYMLP`(*in\_channels=256, hid\_channels=4096, out\_channels=256, num\_layers=2, with\_avg\_pool=True*)

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**(*in\_channels=256, hid\_channels=4096, out\_channels=256, num\_layers=2, with\_avg\_pool=True*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**init\_weights**(*init\_linear='normal'*)

**training:** bool

**class** `easycv.models.selfsup.necks.NonLinearNeckSwav`(*in\_channels, hid\_channels, out\_channels, with\_avg\_pool=True, export=False*)

Bases: `torch.nn.modules.module.Module`

The non-linear neck in byol: fc-synclbn-relu-fc

**\_\_init\_\_**(*in\_channels, hid\_channels, out\_channels, with\_avg\_pool=True, export=False*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(*init\_linear='normal'*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** easycv.models.selfsup.necks.**NonLinearNeckV0**(*in\_channels, hid\_channels, out\_channels,*  
*sync\_bn=False, with\_avg\_pool=True*)

Bases: `torch.nn.modules.module.Module`

The non-linear neck in ODC, fc-bn-relu-dropout-fc-relu

**\_\_init\_\_**(*in\_channels, hid\_channels, out\_channels, sync\_bn=False, with\_avg\_pool=True*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**init\_weights**(*init\_linear='normal'*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** easycv.models.selfsup.necks.**NonLinearNeckV1**(*in\_channels, hid\_channels, out\_channels,*  
*with\_avg\_pool=True*)

Bases: `torch.nn.modules.module.Module`

The non-linear neck in MoCO v2: fc-relu-fc

**\_\_init\_\_**(*in\_channels, hid\_channels, out\_channels, with\_avg\_pool=True*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**init\_weights**(*init\_linear='normal'*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

---

```
class easycv.models.selfsup.necks.NonLinearNeckV2(in_channels, hid_channels, out_channels,
 with_avg_pool=True)
```

Bases: torch.nn.modules.module.Module

The non-linear neck in byol: fc-bn-relu-fc

```
__init__(in_channels, hid_channels, out_channels, with_avg_pool=True)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
init_weights(init_linear='normal')
```

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class easycv.models.selfsup.necks.NonLinearNeckSimCLR(in_channels, hid_channels, out_channels,
 num_layers=2, with_avg_pool=True)
```

Bases: torch.nn.modules.module.Module

SimCLR non-linear neck.

**Structure:** fc(no\_bias)-bn(has\_bias)-[relu-fc(no\_bias)-bn(no\_bias)]. The substructures in [] can be repeated.

For the SimCLR default setting, the repeat time is 1.

**However, PyTorch does not support to specify (weight=True, bias=False).** It only support “affine” including the weight and bias. Hence, the second BatchNorm has bias in this implementation. This is different from the official implementation of SimCLR.

**Since SyncBatchNorm in pytorch<1.4.0 does not support 2D input, the input is expanded to 4D** with shape: (N,C,1,1). I am not sure if this workaround has no bugs. See the pull request here: <https://github.com/pytorch/pytorch/pull/29626>

**Parameters**

- **in\_channels** – input channel number
- **hid\_channels** – hidden channels
- **out\_channels** – output channel number
- **num\_layers** (*int*) – number of fc layers, it is 2 in the SimCLR default setting.
- **with\_avg\_pool** – output with average pooling

```
__init__(in_channels, hid_channels, out_channels, num_layers=2, with_avg_pool=True)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
init_weights(init_linear='normal')
```

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks

---

while the latter silently ignores them.

---

**training:** bool

**class** easycv.models.selfsup.necks.**RelativeLocNeck**(*in\_channels*, *out\_channels*, *sync\_bn=False*,  
*with\_avg\_pool=True*)

Bases: torch.nn.modules.module.Module

Relative patch location neck: fc-bn-relu-dropout

**\_\_init\_\_**(*in\_channels*, *out\_channels*, *sync\_bn=False*, *with\_avg\_pool=True*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(*init\_linear='normal'*)

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**class** easycv.models.selfsup.necks.**MAENeck**(*num\_patches*, *embed\_dim=768*, *patch\_size=16*, *in\_chans=3*,  
*decoder\_embed\_dim=512*, *decoder\_depth=8*,  
*decoder\_num\_heads=16*, *mlp\_ratio=4.0*,  
*norm\_layer=functools.partial(<class*  
*'torch.nn.modules.normalization.LayerNorm'>, eps=1e-06))*

Bases: torch.nn.modules.module.Module

MAE decoder

**Parameters**

- **num\_patches** (*int*) – number of patches from encoder
- **embed\_dim** (*int*) – encoder embedding dimension
- **patch\_size** (*int*) – encoder patch size
- **in\_chans** (*int*) – input image channels
- **decoder\_embed\_dim** (*int*) – decoder embedding dimension
- **decoder\_depth** (*int*) – number of decoder layers
- **decoder\_num\_heads** (*int*) – Parallel attention heads
- **mlp\_ratio** (*float*) – mlp ratio
- **norm\_layer** – type of normalization layer

**\_\_init\_\_**(*num\_patches*, *embed\_dim=768*, *patch\_size=16*, *in\_chans=3*, *decoder\_embed\_dim=512*,  
*decoder\_depth=8*, *decoder\_num\_heads=16*, *mlp\_ratio=4.0*, *norm\_layer=functools.partial(<class*  
*'torch.nn.modules.normalization.LayerNorm'>, eps=1e-06))*

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**training:** bool

**forward**(*x, ids\_restore*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

## easycv.models.selfsup.simclr module

**class** easycv.models.selfsup.simclr.**SimCLR**(*backbone, train\_preprocess=[], neck=None, head=None, pretrained=None*)

Bases: [easycv.models.base.BaseModel](#)

**\_\_init\_\_**(*backbone, train\_preprocess=[], neck=None, head=None, pretrained=None*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(*pretrained=None*)

**forward\_backbone**(*img*)

Forward backbone

**Returns** backbone outputs

**Return type** *x* (tuple)

**forward\_train**(*img, \*\*kwargs*)

Abstract interface for model forward in training

**Parameters**

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward\_test**(*img, \*\*kwargs*)

Abstract interface for model forward in testing

**Parameters**

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward**(*img, mode='train', \*\*kwargs*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**easycv.models.selfsup.swav module**

**class** easycv.models.selfsup.swav.**SWAV**(*backbone*, *train\_preprocess*=[], *neck*=None, *config*=None, *pretrained*=None)

Bases: [easycv.models.base.BaseModel](#)

**\_\_init\_\_**(*backbone*, *train\_preprocess*=[], *neck*=None, *config*=None, *pretrained*=None)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**init\_weights**(*pretrained*=None)

**forward\_backbone**(*img*)

**forward\_train\_model**(*inputs*)

**forward\_train**(*inputs*)

Abstract interface for model forward in training

**Parameters**

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward\_test**(*img*, *\*\*kwargs*)

Abstract interface for model forward in testing

**Parameters**

- **img** (*Tensor*) – image tensor
- **kwargs** (*keyword arguments*) – Specific to concrete implementation.

**forward\_feature**(*img*, *\*\*kwargs*)

Forward backbone

**Returns** feature tensor

**Return type** x (*torch.Tensor*)

**forward**(*img*, *gt\_label*=None, *mode*='train', *extract\_list*=['neck'], *\*\*kwargs*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

**class** easycv.models.selfsup.swav.**MultiPrototypes**(*output\_dim*, *nmb\_prototypes*)

Bases: [torch.nn.modules.module.Module](#)

**\_\_init\_\_**(*output\_dim*, *nmb\_prototypes*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.



---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

`easycv.models.selfsup.swav.distributed_sinkhorn(Q, nmb_iters)`

## 16.1.8 easycv.models.utils package

### Submodules

#### easycv.models.utils.accuracy module

`easycv.models.utils.accuracy.accuracy(pred, target, topk=1)`

##### Parameters

- **pred** – [N x num\_classes]
- **target** – [num\_classes]

**class** `easycv.models.utils.accuracy.Accuracy(topk=1)`

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**(*topk=1*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*pred, target*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

#### easycv.models.utils.activation module

**class** `easycv.models.utils.activation.FReLU(in_channel)`

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**(*in\_channel*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

### `easycv.models.utils.conv_module` module

`easycv.models.utils.conv_module.build_conv_layer(cfg, *args, **kwargs)`

Build convolution layer

**Parameters** `cfg` (*None* or *dict*) – `cfg` should contain: `type` (*str*): identify conv layer type. `layer` `args`: args needed to instantiate a conv layer.

**Returns** created conv layer

**Return type** `layer` (`nn.Module`)

**class** `easycv.models.utils.conv_module.ConvModule`(*in\_channels*, *out\_channels*, *kernel\_size*, *stride*=1, *padding*=0, *dilation*=1, *groups*=1, *bias*='auto', *conv\_cfg*=None, *norm\_cfg*=None, *activation*='relu', *inplace*=True, *order*=('conv', 'norm', 'act'))

Bases: `torch.nn.modules.module.Module`

A conv block that contains conv/norm/activation layers.

**Parameters**

- **`in_channels`** (*int*) – Same as `nn.Conv2d`.
- **`out_channels`** (*int*) – Same as `nn.Conv2d`.
- **`kernel_size`** (*int* or *tuple[int]*) – Same as `nn.Conv2d`.
- **`stride`** (*int* or *tuple[int]*) – Same as `nn.Conv2d`.
- **`padding`** (*int* or *tuple[int]*) – Same as `nn.Conv2d`.
- **`dilation`** (*int* or *tuple[int]*) – Same as `nn.Conv2d`.
- **`groups`** (*int*) – Same as `nn.Conv2d`.
- **`bias`** (*bool* or *str*) – If specified as *auto*, it will be decided by the `norm_cfg`. Bias will be set as *True* if `norm_cfg` is *None*, otherwise *False*.
- **`conv_cfg`** (*dict*) – Config dict for convolution layer.
- **`norm_cfg`** (*dict*) – Config dict for normalization layer.
- **`activation`** (*str* or *None*) – Activation type, “ReLU” by default.
- **`inplace`** (*bool*) – Whether to use inplace mode for activation.
- **`order`** (*tuple[str]*) – The order of conv/norm/activation layers. It is a sequence of “conv”, “norm” and “act”. Examples are (“conv”, “norm”, “act”) and (“act”, “conv”, “norm”).

**`__init__`**(*in\_channels*, *out\_channels*, *kernel\_size*, *stride*=1, *padding*=0, *dilation*=1, *groups*=1, *bias*='auto', *conv\_cfg*=None, *norm\_cfg*=None, *activation*='relu', *inplace*=True, *order*=('conv', 'norm', 'act'))

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**property** `norm`

**`init_weights()`**

**forward**(*x*, *activate=True*, *norm=True*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

### easy cv.models.utils.conv\_ws module

easy cv.models.utils.conv\_ws.**conv\_ws\_2d**(*input*, *weight*, *bias=None*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *eps=1e-05*)

**class** easy cv.models.utils.conv\_ws.**ConvWS2d**(*in\_channels*, *out\_channels*, *kernel\_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *eps=1e-05*)

Bases: torch.nn.modules.conv.Conv2d

**\_\_init\_\_**(*in\_channels*, *out\_channels*, *kernel\_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *eps=1e-05*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**bias:** Optional[torch.Tensor]

**out\_channels:** int

**kernel\_size:** Tuple[int, ...]

**stride:** Tuple[int, ...]

**padding:** Union[str, Tuple[int, ...]]

**dilation:** Tuple[int, ...]

**transposed:** bool

**output\_padding:** Tuple[int, ...]

**groups:** int

**padding\_mode:** str

**weight:** torch.Tensor

**easycv.models.utils.dist\_utils module**

`easycv.models.utils.dist_utils.is_distributed()`

`easycv.models.utils.dist_utils.all_gather(embeddings, labels)`

`easycv.models.utils.dist_utils.all_gather_embeddings_labels(embeddings, labels)`

**class** `easycv.models.utils.dist_utils.DistributedLossWrapper(loss, **kwargs)`

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**(*loss, \*\*kwargs*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward**(*embeddings, labels, \*args, \*\*kwargs*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `easycv.models.utils.dist_utils.DistributedMinerWrapper(miner)`

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**(*miner*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward**(*embeddings, labels, ref\_emb=None, ref\_labels=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**easycv.models.utils.gather\_layer module**

**class** `easycv.models.utils.gather_layer.GatherLayer(*args, **kwargs)`

Bases: `torch.autograd.function.Function`

Gather tensors from all process, supporting backward propagation.

**static forward**(*ctx, input*)

Performs the operation.

This function is to be overridden by all subclasses.

It must accept a context `ctx` as the first argument, followed by any number of arguments (tensors or other types).

The context can be used to store arbitrary data that can be then retrieved during the backward pass. Tensors should not be stored directly on `ctx` (though this is not currently enforced for backward compatibility).

Instead, tensors should be saved either with `ctx.save_for_backward()` if they are intended to be used in backward (equivalently, `vjp`) or `ctx.save_for_forward()` if they are intended to be used for in `jvp`.

**static backward**(*ctx*, \**grads*)

Defines a formula for differentiating the operation with backward mode automatic differentiation (alias to the `vjp` function).

This function is to be overridden by all subclasses.

It must accept a context `ctx` as the first argument, followed by as many outputs as the `forward()` returned (None will be passed in for non tensor outputs of the forward function), and it should return as many tensors, as there were inputs to `forward()`. Each argument is the gradient w.r.t the given output, and each returned value should be the gradient w.r.t. the corresponding input. If an input is not a Tensor or is a Tensor not requiring grads, you can just pass None as a gradient for that input.

The context can be used to retrieve tensors saved during the forward pass. It also has an attribute `ctx.needs_input_grad` as a tuple of booleans representing whether each input needs gradient. E.g., `backward()` will have `ctx.needs_input_grad[0] = True` if the first input to `forward()` needs gradient computed w.r.t. the output.

## easycv.models.utils.init\_weights module

`easycv.models.utils.init_weights.trunc_normal_(tensor, mean=0.0, std=1.0, a=- 2.0, b=2.0)`

## easycv.models.utils.multi\_pooling module

**class** `easycv.models.utils.multi_pooling.GeMPooling`(*p=3*, *eps=1e-06*)

Bases: `torch.nn.modules.module.Module`

GemPooling used for image retrieval  $p = 1$ , avgpooling  $p > 1$  : increases the contrast of the pooled feature map and focuses on the salient features of the image  $p = \text{infinite}$  : spatial max-pooling layer

**\_\_init\_\_**(*p=3*, *eps=1e-06*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward**(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**gem**(*x*, *p=3*, *eps=1e-06*)

**training:** `bool`

**class** `easycv.models.utils.multi_pooling.MultiPooling`(*pool\_type='adaptive'*, *in\_indices=(0)*,  
*backbone='resnet50'*)

Bases: `torch.nn.modules.module.Module`

Pooling layers for features from multiple depth.

```
POOL_PARAMS = {'resnet50': [{'kernel_size': 10, 'stride': 10, 'padding': 4},
{'kernel_size': 16, 'stride': 8, 'padding': 0}, {'kernel_size': 13, 'stride':
5, 'padding': 0}, {'kernel_size': 8, 'stride': 3, 'padding': 0}, {'kernel_size':
6, 'stride': 1, 'padding': 0}]}
```

```
POOL_SIZES = {'resnet50': [12, 6, 4, 3, 2]}
```

```
POOL_DIMS = {'resnet50': [9216, 9216, 8192, 9216, 8192]}
```

```
__init__(pool_type='adaptive', in_indices=(0), backbone='resnet50')
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
training: bool
```

```
class easycv.models.utils.multi_pooling.MultiAvgPooling(pool_type='adaptive', in_indices=(0),
 backbone='resnet50')
```

Bases: torch.nn.modules.module.Module

Pooling layers for features from multiple depth.

```
POOL_PARAMS = {'resnet50': [{'kernel_size': 10, 'stride': 10, 'padding': 4},
 {'kernel_size': 16, 'stride': 8, 'padding': 0}, {'kernel_size': 13, 'stride':
5, 'padding': 0}, {'kernel_size': 8, 'stride': 3, 'padding': 0}, {'kernel_size':
7, 'stride': 1, 'padding': 0}]}
```

```
POOL_SIZES = {'resnet50': [12, 6, 4, 3, 1]}
```

```
POOL_DIMS = {'resnet50': [9216, 9216, 8192, 9216, 2048]}
```

```
__init__(pool_type='adaptive', in_indices=(0), backbone='resnet50')
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
training: bool
```

```
forward(x)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

## easycv.models.utils.norm module

```
class easycv.models.utils.norm.SyncIBN(planes, ratio=0.5, eps=1e-05)
```

Bases: torch.nn.modules.module.Module

Instance-Batch Normalization layer from “Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net” <<https://arxiv.org/pdf/1807.09441.pdf>> :param planes: Number of channels for the input tensor :type planes: int :param ratio: Ratio of instance normalization in the IBN layer :type ratio: float

```
__init__(planes, ratio=0.5, eps=1e-05)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `easycv.models.utils.norm.IBN(planes, ratio=0.5, eps=1e-05)`

Bases: `torch.nn.modules.module.Module`

Instance-Batch Normalization layer from “Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net” <<https://arxiv.org/pdf/1807.09441.pdf>> :param planes: Number of channels for the input tensor :type planes: int :param ratio: Ratio of instance normalization in the IBN layer :type ratio: float

**\_\_init\_\_**(planes, ratio=0.5, eps=1e-05)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward(x)**

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

`easycv.models.utils.norm.build_norm_layer(cfg, num_features, postfix=“”)`

Build normalization layer

**Parameters**

- **cfg** (*dict*) – cfg should contain: type (str): identify norm layer type. layer args: args needed to instantiate a norm layer. requires\_grad (bool): [optional] whether stop gradient updates
- **num\_features** (*int*) – number of channels from input.
- **postfix** (*int*, *str*) – appended into norm abbreviation to create named layer.

**Returns** abbreviation + postfix layer (`nn.Module`): created norm layer

**Return type** name (str)

### easycv.models.utils.ops module

`easycv.models.utils.ops.resize_tensor(input, size=None, scale_factor=None, mode='nearest', align_corners=None, warning=True)`

Resize tensor with F.interpolate.

#### Parameters

- **input** (*Tensor*) – the input tensor.
- **size** (*Tuple[int, int]*) – output spatial size.
- **scale\_factor** (*float or Tuple[float]*) – multiplier for spatial size. If `scale_factor` is a tuple, its length has to match `input.dim()`.
- **mode** (*str*) – algorithm used for upsampling: ‘nearest’ | ‘linear’ | ‘bilinear’ | ‘bicubic’ | ‘trilinear’ | ‘area’. Default: ‘nearest’
- **align\_corners** (*bool*) – Geometrically, we consider the pixels of the input and output as squares rather than points. If set to True, the input and output tensors are aligned by the center points of their corner pixels, preserving the values at the corner pixels.

If set to False, the input and output tensors are aligned by the corner points of their corner pixels, and the interpolation uses edge value padding for out-of-boundary values, making this operation independent of input size when `scale_factor` is kept the same. This only has an effect when mode is ‘linear’, ‘bilinear’, ‘bicubic’ or ‘trilinear’.

### easycv.models.utils.pos\_embed module

`easycv.models.utils.pos_embed.get_2d_sincos_pos_embed(embed_dim, grid_size, cls_token=False)`  
grid\_size: int of the grid height and width return: pos\_embed: [grid\_size\*grid\_size, embed\_dim] or [1+grid\_size\*grid\_size, embed\_dim] (w/ or w/o cls\_token)

`easycv.models.utils.pos_embed.get_2d_sincos_pos_embed_from_grid(embed_dim, grid)`

`easycv.models.utils.pos_embed.get_1d_sincos_pos_embed_from_grid(embed_dim, pos)`  
embed\_dim: output dimension for each position pos: a list of positions to be encoded: size (M,) out: (M, D)

`easycv.models.utils.pos_embed.interpolate_pos_embed(model, checkpoint_model)`

### easycv.models.utils.res\_layer module

`class easycv.models.utils.res_layer.ResLayer(block, num_blocks, in_channels, out_channels, expansion=None, stride=1, avg_down=False, conv_cfg=None, norm_cfg={'type': 'BN'}, **kwargs)`

Bases: `torch.nn.modules.container.Sequential`

ResLayer to build ResNet style backbone. :param block: Residual block used to build ResLayer. :type block: nn.Module :param num\_blocks: Number of blocks. :type num\_blocks: int :param in\_channels: Input channels of this block. :type in\_channels: int :param out\_channels: Output channels of this block. :type out\_channels: int :param expansion: The expansion for BasicBlock/Bottleneck.

If not specified, it will firstly be obtained via `block.expansion`. If the block has no attribute “expansion”, the following default values will be used: 1 for BasicBlock and 4 for Bottleneck. Default: None.

#### Parameters

- **stride** (*int*) – stride of the first block. Default: 1.



- **avg\_down** (*bool*) – Use AvgPool instead of stride conv when downsampling in the bottleneck. Default: False
- **conv\_cfg** (*dict*, *optional*) – dictionary to construct and config conv layer. Default: None
- **norm\_cfg** (*dict*) – dictionary to construct and config norm layer. Default: dict(type='BN')

**\_\_init\_\_** (*block*, *num\_blocks*, *in\_channels*, *out\_channels*, *expansion=None*, *stride=1*, *avg\_down=False*, *conv\_cfg=None*, *norm\_cfg={'type': 'BN'}*, *\*\*kwargs*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

### easycv.models.utils.scale module

**class** easycv.models.utils.scale.**Scale**(*scale=1.0*)

Bases: torch.nn.modules.module.Module

A learnable scale parameter

**\_\_init\_\_** (*scale=1.0*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

### easycv.models.utils.sobel module

**class** easycv.models.utils.sobel.**Sobel**

Bases: torch.nn.modules.module.Module

**\_\_init\_\_** ()

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

## 16.2 Submodules

### 16.3 easycv.models.base module

**class** easycv.models.base.BaseModel

Bases: torch.nn.modules.module.Module

base class for model.

**abstract forward\_train**(img: torch.Tensor, \*\*kwargs) → Dict[str, torch.Tensor]

Abstract interface for model forward in training

**Parameters**

- **img** (Tensor) – image tensor
- **kwargs** (keyword arguments) – Specific to concrete implementation.

**forward\_test**(img: torch.Tensor, \*\*kwargs) → Dict[str, torch.Tensor]

Abstract interface for model forward in testing

**Parameters**

- **img** (Tensor) – image tensor
- **kwargs** (keyword arguments) – Specific to concrete implementation.

**forward**(img, mode='train', \*\*kwargs)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**train\_step**(data, optimizer)

The iteration step during training.

This method defines an iteration step during training, except for the back propagation and optimizer updating, which are done in an optimizer hook. Note that in some complicated cases or models, the whole process including back propagation and optimizer updating is also defined in this method, such as GAN.

**Parameters**

- **data** (dict) – The output of dataloader.
- **optimizer** (torch.optim.Optimizer | dict) – The optimizer of runner is passed to train\_step(). This argument is unused and reserved.

**Returns**

It should contain at least 3 keys: loss, log\_vars, num\_samples.

- **loss** is a tensor for back propagation, which can be a weighted sum of multiple losses.
- **log\_vars** contains all the variables to be sent to the logger.
- **num\_samples** indicates the batch size (when the model is DDP, it means the batch size on each GPU), which is used for averaging the logs.

**Return type** dict

**val\_step**(*data, optimizer*)

The iteration step during validation.

This method shares the same signature as [train\\_step\(\)](#), but used during val epochs. Note that the evaluation after training epochs is not implemented with this method, but an evaluation hook.

**show\_result**(\*\**kwargs*)

Visualize the results.

**training:** bool

## 16.4 easycv.models.builder module

easycv.models.builder.**build**(*cfg, registry, default\_args=None*)

easycv.models.builder.**build\_backbone**(*cfg*)

easycv.models.builder.**build\_neck**(*cfg*)

easycv.models.builder.**build\_memory**(*cfg*)

easycv.models.builder.**build\_head**(*cfg*)

easycv.models.builder.**build\_loss**(*cfg*)

easycv.models.builder.**build\_model**(*cfg*)

## 16.5 easycv.models.modelzoo module

## 16.6 easycv.models.registry module



## EASYCV.UTILS PACKAGE

### 17.1 Submodules

### 17.2 `easycv.utils.alias_multinomial` module

**class** `easycv.utils.alias_multinomial.AliasMethod(probs)`

Bases: `object`

<https://hips.seas.harvard.edu/blog/2013/03/03/the-alias-method-efficient-sampling-with-many-discrete-outcomes/>

**\_\_init\_\_**(*probs*)

Initialize self. See `help(type(self))` for accurate signature.

**cuda**()

**draw**(*N*)

Draw *N* samples from multinomial

### 17.3 `easycv.utils.bbox_util` module

`easycv.utils.bbox_util.bound_limits(v)`

`easycv.utils.bbox_util.bound_limits_for_list(xywh)`

`easycv.utils.bbox_util.xyxy2xywh_with_shape(x, shape)`

`easycv.utils.bbox_util.batched_cxcywh2xyxy_with_shape(bboxes, shape)`

reverse of `xyxy2xywh_with_shape` transform normalized points `[[x_center, y_center, box_w, box_h], ...]` to standard `[[x1, y1, x2, y2], ...]` :param `bboxes`: np.array or tensor like `[[x_center, y_center, box_w, box_h], ...]`, all value is normalized

**Parameters** `shape` – img shape: `[h, w]`

return: np.array or tensor like `[[x1, y1, x2, y2], ...]`

`easycv.utils.bbox_util.batched_xyxy2cxcywh_with_shape(bboxes, shape)`

`easycv.utils.bbox_util.xyxy2xywh_coco(bboxes, offset=0)`

`easycv.utils.bbox_util.xywh2xyxy_coco(bboxes, offset=0)`

`easycv.utils.bbox_util.xyxy2xywh(x)`

`easycv.utils.bbox_util.xywh2xyxy(x)`

`easycv.utils.bbox_util.bbox_iou(box1, box2, x1y1x2y2=True, GIoU=False, DIoU=False, CIoU=False, eps=1e-09)`

`easycv.utils.bbox_util.box_iou(box1, box2)`

Return intersection-over-union (Jaccard index) of boxes. Both sets of boxes are expected to be in (x1, y1, x2, y2) format. :param box1: :type box1: Tensor[N, 4] :param box2: :type box2: Tensor[M, 4]

**Returns**

the NxM matrix containing the pairwise IoU values for every element in boxes1 and boxes2

**Return type** iou (Tensor[N, M])

`easycv.utils.bbox_util.box_candidates(box1, box2, wh_thr=2, ar_thr=20, area_thr=0.1)`

Compute candidate boxes: box1 before augment, box2 after augment, wh\_thr (pixels), aspect\_ratio\_thr, area\_ratio

`easycv.utils.bbox_util.clip_coords(boxes: torch.Tensor, img_shape: Tuple[int, int]) → None`

Clip bounding xyxy bounding boxes to image shape

**Parameters**

- **boxes** – tensor with shape Nx4 (x1,y1,x2,y2)
- **img\_shape** – image size tuple, (height, width)

`easycv.utils.bbox_util.scale_coords(img1_shape: Tuple[int, int], coords: torch.Tensor, img0_shape: Tuple[int, int], ratio_pad: Optional[Tuple[Tuple[float, float], Tuple[float, float]]] = None)`

## 17.4 easycv.utils.checkpoint module

`easycv.utils.checkpoint.load_checkpoint(model, filename, map_location='cpu', strict=False, logger=None)`

Load checkpoint from a file or URI.

**Parameters**

- **model** (*Module*) – Module to load checkpoint.
- **filename** (*str*) – Accept local filepath, URL, torchvision://xxx, open-mmlab://xxx. Please refer to docs/model\_zoo.md for details.
- **map\_location** (*str*) – Same as torch.load().
- **strict** (*bool*) – Whether to allow different params for the model and checkpoint.
- **logger** (*logging.Logger* or *None*) – The logger for error message.

**Returns** The loaded checkpoint.

**Return type** dict or OrderedDict

`easycv.utils.checkpoint.save_checkpoint(model, filename, optimizer=None, meta=None)`

Save checkpoint to file.

The checkpoint will have 3 fields: meta, state\_dict and optimizer. By default meta will contain version and time info.

**Parameters**

- **model** (*Module*) – Module whose params are to be saved.
- **filename** (*str*) – Checkpoint filename.

- **optimizer** (Optimizer, optional) – Optimizer to be saved.
- **meta** (*dict*, *optional*) – Metadata to be saved in checkpoint.

## 17.5 easycv.utils.collect module

`easycv.utils.collect.nondist_forward_collect(func, data_loader, length)`

Forward and collect network outputs.

This function performs forward propagation and collects outputs. It can be used to collect results, features, losses, etc.

### Parameters

- **func** (*function*) – The function to process data. The output must be a dictionary of CPU tensors.
- **length** (*int*) – Expected length of output arrays.

**Returns** The concatenated outputs.

**Return type** `results_all` (`dict(np.ndarray)`)

`easycv.utils.collect.dist_forward_collect(func, data_loader, rank, length, ret_rank=-1)`

Forward and collect network outputs in a distributed manner.

This function performs forward propagation and collects outputs. It can be used to collect results, features, losses, etc.

### Parameters

- **func** (*function*) – The function to process data. The output must be a dictionary of CPU tensors.
- **rank** (*int*) – This process id.
- **length** (*int*) – Expected length of output arrays.
- **ret\_rank** (*int*) – The process that returns. Other processes will return None.

**Returns** The concatenated outputs.

**Return type** `results_all` (`dict(np.ndarray)`)

## 17.6 easycv.utils.collect\_env module

`easycv.utils.collect_env.collect_env()`

## 17.7 easycv.utils.config\_tools module

`easycv.utils.config_tools.traverse_replace(d, key, value)`

`easycv.utils.config_tools.check_base_cfg_path(base_cfg_name='configs/base.py', ori_filename=None)`

`easycv.utils.config_tools.mmcv_file2dict_raw(ori_filename)`

`easycv.utils.config_tools.mmcv_file2dict_base(ori_filename)`

`easycv.utils.config_tools.mmcv_config_fromfile(ori_filename)`

`easycv.utils.config_tools.get_config_class_value(cfg_dict, ori_key, dict_mem_helper)`

`easycv.utils.config_tools.config_dict_edit(ori_cfg_dict, cfg_dict, reg, dict_mem_helper)`  
edit `{configs.variables}` in config dict to solve dependencies in config

`ori_cfg_dict`: to find the true value of `{configs.variables}` `cfg_dict`: for find leafs of dict by recursive `reg`: Regular expression pattern for find all `{configs.variables}` in leafs of dict `dict_mem_helper`: to store the true value of `{configs.variables}` which have been found

`easycv.utils.config_tools.rebuild_config(cfg, user_config_params)`  
# rebuild config by user config params, modify config by user config params & replace `{configs.variables}` by true value

return: Config

`easycv.utils.config_tools.validate_export_config(cfg)`

## 17.8 easycv.utils.constant module

## 17.9 easycv.utils.dist\_utils module

`easycv.utils.dist_utils.is_master()`

`easycv.utils.dist_utils.local_rank()`

`easycv.utils.dist_utils.dist_zero_exec(rank=0)`

`easycv.utils.dist_utils.get_num_gpu_per_node()`  
get number of gpu per node

`easycv.utils.dist_utils.barrier()`

`easycv.utils.dist_utils.is_parallel(model)`

`easycv.utils.dist_utils.obj2tensor(pyobj, device='cuda')`  
Serialize picklable python object to tensor.

`easycv.utils.dist_utils.tensor2obj(tensor)`  
Deserialize tensor to picklable python object.

`easycv.utils.dist_utils.all_reduce_dict(py_dict, op='sum', group=None, to_float=True)`  
Apply all reduce function for python dict object.

The code is modified from [https://github.com/Megvii-BaseDetection/YOLOX/blob/main/yolox/utils/allreduce\\_norm.py](https://github.com/Megvii-BaseDetection/YOLOX/blob/main/yolox/utils/allreduce_norm.py).

NOTE: make sure that `py_dict` in different ranks has the same keys and the values should be in the same shape.

### Parameters

- **py\_dict** (*dict*) – Dict to be applied all reduce op.
- **op** (*str*) – Operator, could be 'sum' or 'mean'. Default: 'sum'
- **group** (*torch.distributed.group*, optional) – Distributed group, Default: None.
- **to\_float** (*bool*) – Whether to convert all values of dict to float. Default: True.

**Returns** reduced python dict object.

**Return type** OrderedDict



## 17.10 easycv.utils.eval\_utils module

`easycv.utils.eval_utils.generate_best_metric_name(evaluate_type, dataset_name, metric_names)`

Generate best metric name for different evaluator / different dataset / different metric\_names evaluate\_type: str  
dataset\_name: None or str metric\_names: None str or list[str] or tuple(str)

**Returns** list[str]

## 17.11 easycv.utils.flops\_counter module

`easycv.utils.flops_counter.get_model_info(model, input_size, model_config, logger)`

get\_model\_info, check model parameters and Gflops

`easycv.utils.flops_counter.get_model_complexity_info(model, input_res, print_per_layer_stat=True, as_strings=True, input_constructor=None, ost=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

`easycv.utils.flops_counter.flops_to_string(flops, units='GMac', precision=2)`

`easycv.utils.flops_counter.params_to_string(params_num)`

converting number to string

**Parameters** `params_num` (float) – number

**Returns** str number

```
>>> params_to_string(1e9)
'1000.0 M'
>>> params_to_string(2e5)
'200.0 k'
>>> params_to_string(3e-9)
'3e-09'
```

`easycv.utils.flops_counter.print_model_with_flops(model, units='GMac', precision=3, ost=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

`easycv.utils.flops_counter.get_model_parameters_number(model)`

`easycv.utils.flops_counter.add_flops_counting_methods(net_main_module)`

`easycv.utils.flops_counter.compute_average_flops_cost(self)`

A method that will be available after add\_flops\_counting\_methods() is called on a desired net object. Returns current mean flops consumption per image.

`easycv.utils.flops_counter.start_flops_count(self)`

A method that will be available after add\_flops\_counting\_methods() is called on a desired net object. Activates the computation of mean flops consumption per image. Call it before you run the network.

`easycv.utils.flops_counter.stop_flops_count(self)`

A method that will be available after add\_flops\_counting\_methods() is called on a desired net object. Stops computing the mean flops consumption per image. Call whenever you want to pause the computation.

`easycv.utils.flops_counter.reset_flops_count(self)`

A method that will be available after add\_flops\_counting\_methods() is called on a desired net object. Resets statistics computed so far.

`easycv.utils.flops_counter.add_flops_mask(module, mask)`

```
easycv.utils.flops_counter.remove_flops_mask(module)
easycv.utils.flops_counter.is_supported_instance(module)
easycv.utils.flops_counter.empty_flops_counter_hook(module, input, output)
easycv.utils.flops_counter.upsample_flops_counter_hook(module, input, output)
easycv.utils.flops_counter.relu_flops_counter_hook(module, input, output)
easycv.utils.flops_counter.linear_flops_counter_hook(module, input, output)
easycv.utils.flops_counter.pool_flops_counter_hook(module, input, output)
easycv.utils.flops_counter.bn_flops_counter_hook(module, input, output)
easycv.utils.flops_counter.gn_flops_counter_hook(module, input, output)
easycv.utils.flops_counter.deconv_flops_counter_hook(conv_module, input, output)
easycv.utils.flops_counter.conv_flops_counter_hook(conv_module, input, output)
easycv.utils.flops_counter.batch_counter_hook(module, input, output)
easycv.utils.flops_counter.add_batch_counter_variables_or_reset(module)
easycv.utils.flops_counter.add_batch_counter_hook_function(module)
easycv.utils.flops_counter.remove_batch_counter_hook_function(module)
easycv.utils.flops_counter.add_flops_counter_variable_or_reset(module)
easycv.utils.flops_counter.add_flops_counter_hook_function(module)
easycv.utils.flops_counter.remove_flops_counter_hook_function(module)
easycv.utils.flops_counter.add_flops_mask_variable_or_reset(module)
```

## 17.12 easycv.utils.gather module

```
easycv.utils.gather.gather_tensors(input_array)
easycv.utils.gather.gather_tensors_batch(input_array, part_size=100, ret_rank=-1)
```

## 17.13 easycv.utils.json\_utils module

Utilities for dealing with writing json strings.

json\_utils wraps json.dump and json.dumps so that they can be used to safely control the precision of floats when writing to json strings or files.

```
class easycv.utils.json_utils.MyEncoder(*, skipkeys=False, ensure_ascii=True, check_circular=True,
 allow_nan=True, sort_keys=False, indent=None,
 separators=None, default=None)
```

Bases: json.encoder.JSONEncoder

**default**(o)

Implement this method in a subclass such that it returns a serializable object for o, or calls the base implementation (to raise a TypeError).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
 try:
 iterable = iter(o)
 except TypeError:
 pass
 else:
 return list(iterable)
 # Let the base class default method raise the TypeError
 return JSONEncoder.default(self, o)
```

**iterencode**(o, \_one\_shot=False)

Encode the given object and yield each string representation as available.

For example:

```
for chunk in JSONEncoder().iterencode(bigobject):
 mysocket.write(chunk)
```

**easycv.utils.json\_utils.dump**(obj, fid, float\_digits=-1, \*\*params)

Wrapper of json.dump that allows specifying the float precision used.

**Parameters**

- **obj** – The object to dump.
- **fid** – The file id to write to.
- **float\_digits** – The number of digits of precision when writing floats out.
- **\*\*params** – Additional parameters to pass to json.dumps.

**easycv.utils.json\_utils.dumps**(obj, float\_digits=-1, \*\*params)

Wrapper of json.dumps that allows specifying the float precision used.

**Parameters**

- **obj** – The object to dump.
- **float\_digits** – The number of digits of precision when writing floats out.
- **\*\*params** – Additional parameters to pass to json.dumps.

**Returns** JSON string representation of obj.

**Return type** output

**easycv.utils.json\_utils.compat\_dumps**(data, float\_digits=-1)

handle json dumps chinese and numpy data :param data python data structure: :param float\_digits: The number of digits of precision when writing floats out.

**Returns**

json str, in python2 , the str is encoded with utf8 in python3, the str is unicode  
type(python3 str)

**easycv.utils.json\_utils.PrettyParams**(\*\*params)

Returns parameters for use with Dump and Dumps to output pretty json.

**Example usage:** `json\_str = json\_utils.Dumps(obj, \*\*json\_utils.PrettyParams())`  
`json\_str = json\_utils.Dumps(`

obj, \*\*json\_utils.PrettyParams(allow\_nans=False))`

**Parameters** **\*\*params** – Additional params to pass to json.dump or json.dumps.

**Returns**

Parameters that are compatible with `json_utils.Dump` and `json_utils.Dumps`.

**Return type** `params`

## 17.14 easycv.utils.logger module

`easycv.utils.logger.get_root_logger(log_file=None, log_level=20)`

Get the root logger.

The logger will be initialized if it has not been initialized. By default a `StreamHandler` will be added. If `log_file` is specified, a `FileHandler` will also be added. The name of the root logger is the top-level package name, e.g., “easycv”.

**Parameters**

- **log\_file** (*str* / *None*) – The log filename. If specified, a `FileHandler` will be added to the root logger.
- **log\_level** (*int*) – The root logger level. Note that only the process of rank 0 is affected, while other processes will set the level to “Error” and be silent most of the time.

**Returns** The root logger.

**Return type** `logging.Logger`

`easycv.utils.logger.print_log(msg, logger=None, level=20)`

Print a log message.

**Parameters**

- **msg** (*str*) – The message to be logged.
- **logger** (*logging.Logger* / *str* / *None*) – The logger to be used. Some special loggers are: - “root”: the root logger obtained with `get_root_logger()`. - “silent”: no message will be printed. - *None*: The `print()` method will be used to print log messages.
- **level** (*int*) – Logging level. Only available when `logger` is a `Logger` object or “root”.

## 17.15 easycv.utils.metric\_distance module

`easycv.utils.metric_distance.LpDistance(query_emb, ref_emb, p=2)`

**Input:** `query_emb`: [n, dims] tensor `ref_emb`: [m, dims] tensor `p`: p normalize

**Output:** `distance_matrix`: [n, m] tensor

$$\text{distance\_matrix}_{i,j} = (\sum_k (a_{i,k}^p - b_{j,k}^p))^{1/p}$$

`easycv.utils.metric_distance.DotproductSimilarity(query_emb, ref_emb)`

`easycv.utils.metric_distance.CosineSimilarity(query_emb, ref_emb)`

**Input:** `query_emb`: [n, dims] tensor `ref_emb`: [m, dims] tensor

**Output:** `distance_matrix`: [n, m] tensor

## 17.16 easycv.utils.misc module

`easycv.utils.misc.tensor2imgs(tensor, mean=(0, 0, 0), std=(1, 1, 1), to_rgb=True)`

`easycv.utils.misc.multi_apply(func, *args, **kwargs)`

`easycv.utils.misc.unmap(data, count, inds, fill=0)`

Unmap a subset of item (data) back to the original set of items (of size count)

## 17.17 easycv.utils.preprocess\_function module

`easycv.utils.preprocess_function.bninceptionPre(image, mean=[104, 117, 128], std=[1, 1, 1])`

### Parameters

- **image** – pytorch Image tensor from PIL (range 0~1), bgr format
- **mean** – norm mean
- **std** – norm val

**Returns** A image norm in 0~255, rgb format

`easycv.utils.preprocess_function.randomErasing(image, probability=0.5, sl=0.02, sh=0.2, r1=0.3, mean=[0.4914, 0.4822, 0.4465])`

`easycv.utils.preprocess_function.solarize(tensor, threshold=0.5, apply_prob=0.2)`  
tensor : pytorch tensor

`easycv.utils.preprocess_function.gaussianBlurDynamic(image, apply_prob=0.5)`

`easycv.utils.preprocess_function.gaussianBlur(image, kernel_size=22, apply_prob=0.5)`

`easycv.utils.preprocess_function.randomGrayScale(image, apply_prob=0.2)`

`easycv.utils.preprocess_function.mixUp(image, alpha=0.2)`

`easycv.utils.preprocess_function.mixUpCls(data, alpha=0.2)`

## 17.18 easycv.utils.profiling module

`easycv.utils.profiling.profile_time(trace_name, name, enabled=True, stream=None, end_stream=None)`  
Print time spent by CPU and GPU.

Useful as a temporary context manager to find sweet spots of code suitable for async implementation.

`easycv.utils.profiling.benchmark_torch_function(iters, f, *args)`

`easycv.utils.profiling.time_synchronized()`

## 17.19 easycv.utils.py\_util module

`easycv.utils.py_util.copy_attr(a, b, include=(), exclude=())`

`easycv.utils.py_util.get_parent_path(path: str)`  
get parent path, support oss-style path

## 17.20 easycv.utils.registry module

**class** `easycv.utils.registry.Registry(name)`

Bases: `object`

**\_\_init\_\_**(name)

Initialize self. See help(type(self)) for accurate signature.

**property** `name`

**property** `module_dict`

**get**(key)

**register\_module**(cls=None, force=False)

`easycv.utils.registry.build_from_cfg(cfg, registry, default_args=None)`

Build a module from config dict.

**Parameters**

- **cfg** (*dict*) – Config dict. It should at least contain the key “type”.
- **registry** (*Registry*) – The registry to search the type from.
- **default\_args** (*dict*, *optional*) – Default initialization arguments.

**Returns** The constructed object.

**Return type** `obj`

## 17.21 easycv.utils.test\_util module

Contains functions which are convenient for unit testing.

`easycv.utils.test_util.get_tmp_dir()`

`easycv.utils.test_util.clear_all_tmp_dirs()`

`easycv.utils.test_util.replace_data_for_test(cfg)`

replace real data with test data

**Parameters** `cfg` – Config object

`easycv.utils.test_util.RunAsSubprocess(f)`

`easycv.utils.test_util.clean_up(test_dir)`

`easycv.utils.test_util.run_in_subprocess(cmd)`

`easycv.utils.test_util.dist_exec_wrapper(cmd, nproc_per_node, node_rank=0, nnodes=1, port='29527',  
addr='127.0.0.1', python_path=None)`

donot forget init dist in your function or script of cmd ``python from mmcv.runner import init_dist  
init_dist(launcher='pytorch')``

```
easycv.utils.test_util.is_port_used(port, host='127.0.0.1')
easycv.utils.test_util.get_random_port()
easycv.utils.test_util.pseudo_dist_init()
```

## 17.22 easycv.utils.user\_config\_params\_utils module

```
easycv.utils.user_config_params_utils.check_value_type(replacement, original)
 convert replacement's type to original's type, support converting str to int or float or list or tuple
```





## EASYCV PACKAGE

### 18.1 Subpackages

#### 18.1.1 easycv.file package

##### Submodules

##### easycv.file.base module

```
class easycv.file.base.IOBase
 Bases: object

 static register(options)
 open(path: str, mode: str = 'r')
 exists(path: str) → bool
 move(src: str, dst: str)
 copy(src: str, dst: str)
 copytree(src: str, dst: str)
 makedirs(path: str, exist_ok=True)
 remove(path: str)
 rmtree(path: str)
 listdir(path: str, recursive=False, full_path=False, contains=None)
 isdir(path: str) → bool
 isfile(path: str) → bool
 abspath(path: str) → str
 last_modified(path: str) → datetime.datetime
 last_modified_str(path: str) → str
 size(path: str) → int
 md5(path: str) → str
 re_remote = re.compile('(oss|https?)://')
 islocal(path: str) → bool
 is_writable(path)
```

```
class easycv.file.base.IOLocal
 Bases: easycv.file.base.IOBase

 open(path, mode='r')
 exists(path)
 move(src, dst)
 copy(src, dst)
 copytree(src, dst)
 makedirs(path, exist_ok=True)
 remove(path)
 rmtree(path)

 listdir(path, recursive=False, full_path=False, contains: Optional[Union[str, List[str]]] = None)
 isdir(path)
 isfile(path)
 glob(path)
 abspath(path)
 last_modified(path)
 last_modified_str(path)
 size(path: str) → int
```

### **easycv.file.file\_io module**

```
easycv.file.file_io.set_oss_env(ak_id: str, ak_secret: str, hosts: Union[str, List[str]], buckets: Union[str,
 List[str]])
```

```
class easycv.file.file_io.IO(max_retry=10)
```

Bases: *easycv.file.base.IOLocal*

IO module to support both local and oss io. If access oss file, you need to authorize OSS, please refer to *IO.access\_oss*.

```
__init__(max_retry=10)
```

Initialize self. See help(type(self)) for accurate signature.

```
access_oss(ak_id: str = "", ak_secret: str = "", hosts: Union[str, List[str]] = "", buckets: Union[str, List[str]]
 = "")
```

If access oss file, you need to authorize OSS as follows:

Method1: from easycv.file import io io.access\_oss(

```
 ak_id='your_accesskey_id', ak_secret='your_accesskey_secret', hosts='your endpoint' or
 ['your endpoint1', 'your endpoint2'], buckets='your bucket' or ['your bucket1', 'your
 bucket2'])
```

**Method2:** Add oss config to your local file *~/.ossutilconfig*, as follows: More oss config information, please refer to: [https://help.aliyun.com/document\\_detail/120072.html](https://help.aliyun.com/document_detail/120072.html) `''' [Credentials]`

```
language = CH endpoint = your endpoint accessKeyID = your_accesskey_id accessKey-
Secret = your_accesskey_secret
```

[**Bucket-Endpoint**] bucket1 = endpoint1 bucket2 = endpoint2

''' Then run the following command, the config file will be read by default to authorize oss.

```
from easycv.file import io
io.access_oss()
```

**open**(*full\_path*, *mode*='r')

Same usage as the python build-in *open*. Support local path and oss path.

### Example

```
from easycv.file import io
io.access_oss(your oss config) # only oss file need, refer to IO.access_oss #
Write something to a oss file. with io.open('oss://bucket_name/demo.txt', 'w') as f:
```

```
f.write("test")
```

```
Read from a oss file. with io.open('oss://bucket_name/demo.txt', 'r') as f:
```

```
print(f.read())
```

**Parameters** **full\_path** – absolute oss path

**exists**(*path*)

Whether the file exists, same usage as *os.path.exists*. Support local path and oss path.

### Example

```
from easycv.file import io
io.access_oss(your oss config) # only oss file need, refer to IO.access_oss
ret = io.exists('oss://bucket_name/dir')
print(ret)
```

**Parameters** **path** – oss path or local path

**move**(*src*, *dst*)

Move src to dst, same usage as *shutil.move*. Support local path and oss path.

### Example

```
from easycv.file import io
io.access_oss(your oss config) # only oss file need, refer to IO.access_oss
move oss file to local
io.move('oss://bucket_name/file.txt', '/your/local/path/file.txt') # move
oss file to oss
io.move('oss://bucket_name/file.txt', 'oss://bucket_name/file.txt') # move
local file to oss
io.move('/your/local/file.txt', 'oss://bucket_name/file.txt') # move
directory
io.move('oss://bucket_name/dir1', 'oss://bucket_name/dir2')
```

**Parameters**

- **src** – oss path or local path
- **dst** – oss path or local path

**safe\_copy**(*src*, *dst*, *try\_max*=3)

oss always bug, we need *safe\_copy*!

**copy**(*src*, *dst*)

Copy a file from src to dst. Same usage as *shutil.copyfile*. If you want to copy a directory, please use *easycv.io.copypath*

### Example

```
from easycv.file import io io.access_oss(your oss config) # only oss file need, refer to IO.access_oss #
Copy a file from local to oss: io.copy('/your/local/file.txt', 'oss://bucket/dir/file.txt')

Copy a oss file to local: io.copy('oss://bucket/dir/file.txt', '/your/local/file.txt')

Copy a file from oss to oss:: io.copy('oss://bucket/dir/file.txt', 'oss://bucket/dir/file2.txt')
```

#### Parameters

- **src** – oss path or local path
- **dst** – oss path or local path

#### **copytree**(*src, dst*)

Copy files recursively from src to dst. Same usage as *shutil.copytree*. If you want to copy a file, please use *easycv.io.copy*.

### Example

```
from easycv.file import io io.access_oss(your oss config) # only oss file need, refer to IO.access_oss # copy
files from local to oss io.copytree(src='/your/local/dir1', dst='oss://bucket_name/dir2') # copy files from
oss to local io.copytree(src='oss://bucket_name/dir2', dst='/your/local/dir1') # copy files from oss to oss
io.copytree(src='oss://bucket_name/dir1', dst='oss://bucket_name/dir2')
```

#### Parameters

- **src** – oss path or local path
- **dst** – oss path or local path

#### **listdir**(*path, recursive=False, full\_path=False, contains: Optional[Union[str, List[str]]] = None*)

List all objects in path. Same usage as *os.listdir*.

### Example

```
from easycv.file import io io.access_oss(your oss config) # only oss file need, refer to IO.access_oss ret =
io.listdir('oss://bucket/dir', recursive=True) print(ret)
```

#### Parameters

- **path** – local file path or oss path.
- **recursive** – If False, only list the top level objects. If True, recursively list all objects.
- **full\_path** – if full path, return files with path prefix.
- **contains** – substr to filter list files.

return: A list of path.

#### **remove**(*path*)

Remove a file or a directory recursively. Same usage as *os.remove* or *shutil.rmtree*.

### Example

```
from easycv.file import io io.access_oss(your oss config) # only oss file need, refer to IO.access_oss #
Remove a oss file io.remove('oss://bucket_name/file.txt')
```

```
Remove a oss directory io.remove('oss://bucket_name/dir/')
```

**Parameters** **path** – local or oss path, file or directory

**rmtree**(*path*)

Remove directory recursively, same usage as *shutil.rmtree*

### Example

```
from easycv.file import io io.access_oss(your oss config) # only oss file need, refer to IO.access_oss
io.remove('oss://bucket_name/dir_name') # Or io.remove('oss://bucket_name/dir_name/')
```

**Parameters** **path** – oss path

**makedirs**(*path*, *exist\_ok=True*)

Create directories recursively, same usage as *os.makedirs*

### Example

```
from easycv.file import io io.access_oss(your oss config) # only oss file need, refer to IO.access_oss
io.makedirs('oss://bucket/new_dir/')
```

**Parameters** **path** – local or oss dir path

**isdir**(*path*)

Return whether a path is directory, same usage as *os.path.isdir*

### Example

```
from easycv.file import io io.access_oss(your oss config) # only oss file need, refer to IO.access_oss
io.isdir('oss://bucket/dir/')
```

**Parameters** **path** – local or oss path

Return: bool, True or False.

**isfile**(*path*)

Return whether a path is file object, same usage as *os.path.isfile*

### Example

```
from easycv.file import io io.access_oss(your oss config) # only oss file need, refer to IO.access_oss
io.isfile('oss://bucket/file.txt')
```

**Parameters** **path** – local or oss path

Return: bool, True or False.

**glob**(*file\_path*)

Return a list of paths matching a pathname pattern. .. rubric:: Example

```
from easycv.file import io io.access_oss(your oss config) # only oss file need, refer to IO.access_oss
io.glob('oss://bucket/dir/*.txt')
```

**Parameters** `path` – local or oss file pattern

Return: list, a list of paths.

**abspath**(`path`)

**authorize**(`path`)

**last\_modified**(`path`)

**last\_modified\_str**(`path`)

**size**(`path: str`) → int

Get the size of file path, same usage as `os.path.getsize`

### Example

```
from easycv.file import io
io.access_oss(your oss config) # only oss file need, refer to IO.access_oss size
= io.size('oss://bucket/file.txt')
print(size)
```

**Parameters** `path` – local or oss path.

Return: size of file in bytes

**class** `easycv.file.file_io.OSSFile(bucket, path, position=0)`

Bases: object

**\_\_init\_\_**(`bucket, path, position=0`)

Initialize self. See help(type(self)) for accurate signature.

**write**(`content`)

**flush**(`retry=0`)

**close**()

**seek**(`position`)

**class** `easycv.file.file_io.BinaryOSSFile(bucket, path)`

Bases: object

**\_\_init\_\_**(`bucket, path`)

Initialize self. See help(type(self)) for accurate signature.

**class** `easycv.file.file_io.NullContextWrapper(obj)`

Bases: object

**\_\_init\_\_**(`obj`)

Initialize self. See help(type(self)) for accurate signature.

### easycv.file.utils module

`easycv.file.utils.create_namedtuple(**kwargs)`

`easycv.file.utils.is_oss_path(s)`

`easycv.file.utils.get_oss_config()`

Get oss config file from env `OSS_CONFIG_FILE`, default file is `~/ossutilconfig`.

`easycv.file.utils.oss_progress(desc)`

`easycv.file.utils.ignore_oss_error(msg="")`

`easycv.file.utils.mute_stderr()`

## 18.1.2 easycv.runner package

### Submodules

#### easycv.runner.ev\_runner module

**class** easycv.runner.ev\_runner.EVRunner(*model, batch\_processor=None, optimizer=None, work\_dir=None, logger=None, meta=None*)

Bases: mmcv.runner.epoch\_based\_runner.EpochBasedRunner

**\_\_init\_\_**(*model, batch\_processor=None, optimizer=None, work\_dir=None, logger=None, meta=None*)

Epoch Runner for easycv, add support for oss IO and file sync.

#### Parameters

- **model** (*torch.nn.Module*) – The model to be run.
- **batch\_processor** (*callable*) – A callable method that process a data batch. The interface of this method should be *batch\_processor(model, data, train\_mode) -> dict*
- **optimizer** (*dict or torch.optim.Optimizer*) – It can be either an optimizer (in most cases) or a dict of optimizers (in models that requires more than one optimizer, e.g., GAN).
- **work\_dir** (*str, optional*) – The working directory to save checkpoints and logs. Defaults to None.
- **logger** (*logging.Logger*) – Logger used during training. Defaults to None. (The default value is just for backward compatibility)
- **meta** (*dict | None*) – A dict records some import information such as environment info and seed, which will be logged in logger hook. Defaults to None.

**run\_iter**(*data\_batch, train\_mode, \*\*kwargs*)

process for each iteration.

#### Parameters

- **data\_batch** – Batch of dict of data.
- **train\_model** (*bool*) – If set True, run training step else validation step.

**train**(*data\_loader, \*\*kwargs*)

Training process for one epoch which will iterate through all training data and call hooks at different stages.

**Parameters** **data\_loader** – data loader object for training

**val**(*data\_loader, \*\*kwargs*)

Validation step which Deprecated, using evaluation hook instead.

**save\_checkpoint**(*out\_dir, filename\_tmpl='epoch\_{}.pth', save\_optimizer=True, meta=None, create\_symlink=True*)

Save checkpoint to file.

#### Parameters

- **out\_dir** – Directory where checkpoint files are to be saved.
- **filename\_tmpl** (*str, optional*) – Checkpoint filename pattern.
- **save\_optimizer** (*bool, optional*) – save optimizer state.
- **meta** (*dict, optional*) – Metadata to be saved in checkpoint.

**current\_lr()**

Get current learning rates.

**Returns**

**Current learning rates of all** param groups. If the runner has a dict of optimizers, this method will return a dict.

**Return type** list[float] | dict[str, list[float]]

**load\_checkpoint**(filename, map\_location=device(type='cpu'), strict=False, logger=None)

Load checkpoint from a file or URL.

**Parameters**

- **filename** (str) – Accept local filepath, URL, torchvision://xxx, open-mmlab://xxx, oss://xxx. Please refer to docs/source/model\_zoo.md for details.
- **map\_location** (str) – Same as torch.load().
- **strict** (bool) – Whether to allow different params for the model and checkpoint.
- **logger** (logging.Logger or None) – The logger for error message.

**Returns** The loaded checkpoint.

**Return type** dict or OrderedDict

**resume**(checkpoint, resume\_optimizer=True, map\_location='default')

Resume state dict from checkpoint.

**Parameters**

- **checkpoint** – Checkpoint path
- **resume\_optimizer** – Whether to resume optimizer state
- **map\_location** (str) – Same as torch.load().

## 18.1.3 easycv.toolkit package

### Subpackages

easycv.toolkit.prune package

### Submodules

easycv.toolkit.prune.prune\_utils module

easycv.toolkit.quantize package

### Submodules

easycv.toolkit.quantize.quantize\_utils module

easycv.toolkit.quantize.quantize\_utils.calib(model, data\_loader)



## 18.2 Submodules

### 18.3 easycv.version module



---

CHAPTER  
**NINETEEN**

---

**EASYCV**



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### e

- easycv, 299
- easycv.apis, 37
- easycv.apis.export, 37
- easycv.apis.test, 37
- easycv.apis.train, 38
- easycv.apis.train\_misc, 39
- easycv.core, 155
- easycv.core.evaluation, 155
- easycv.core.evaluation.ap, 156
- easycv.core.evaluation.auc\_eval, 157
- easycv.core.evaluation.base\_evaluator, 157
- easycv.core.evaluation.builder, 157
- easycv.core.evaluation.classification\_eval, 157
- easycv.core.evaluation.coco\_evaluation, 158
- easycv.core.evaluation.coco\_tools, 160
- easycv.core.evaluation.custom\_cocotools, 155
- easycv.core.evaluation.custom\_cocotools.cocoeval, 155
- easycv.core.evaluation.faceid\_pair\_eval, 167
- easycv.core.evaluation.metric\_registry, 168
- easycv.core.evaluation.mse\_eval, 168
- easycv.core.evaluation.retrival\_topk\_eval, 169
- easycv.core.evaluation.top\_down\_eval, 169
- easycv.core.optimizer, 172
- easycv.core.optimizer.lars, 172
- easycv.core.optimizer.ranger, 173
- easycv.core.post\_processing, 173
- easycv.core.post\_processing.nms, 177
- easycv.core.post\_processing.pose\_transforms, 178
- easycv.core.standard\_fields, 184
- easycv.core.visualization, 181
- easycv.core.visualization.image, 182
- easycv.datasets, 41
- easycv.datasets.builder, 131
- easycv.datasets.classification, 41
- easycv.datasets.classification.data\_sources, 42
- easycv.datasets.classification.data\_sources.class\_list, 46
- easycv.datasets.classification.data\_sources.fashiongen\_h5, 47
- easycv.datasets.classification.data\_sources.image\_list, 47
- easycv.datasets.classification.data\_sources.imagenet\_tfrec, 48
- easycv.datasets.classification.data\_sources.utils, 48
- easycv.datasets.classification.odps, 64
- easycv.datasets.classification.pipelines, 50
- easycv.datasets.classification.pipelines.auto\_augment, 54
- easycv.datasets.classification.pipelines.transform, 63
- easycv.datasets.classification.raw, 64
- easycv.datasets.detection, 65
- easycv.datasets.detection.data\_sources, 67
- easycv.datasets.detection.data\_sources.coco, 71
- easycv.datasets.detection.data\_sources.pai\_format, 72
- easycv.datasets.detection.data\_sources.raw, 73
- easycv.datasets.detection.data\_sources.utils, 74
- easycv.datasets.detection.data\_sources.voc, 74
- easycv.datasets.detection.mix, 93
- easycv.datasets.detection.pipelines, 75
- easycv.datasets.detection.pipelines.mm\_transforms, 84
- easycv.datasets.detection.raw, 95
- easycv.datasets.loader, 96
- easycv.datasets.loader.build\_loader, 97
- easycv.datasets.loader.sampler, 98
- easycv.datasets.pose, 100
- easycv.datasets.pose.data\_sources, 100
- easycv.datasets.pose.data\_sources.coco, 102
- easycv.datasets.pose.data\_sources.top\_down, 102

[103](#)  
[easycv.datasets.pose.pipelines, 103](#)  
[easycv.datasets.pose.pipelines.transforms, 106](#)  
[easycv.datasets.pose.top\\_down, 109](#)  
[easycv.datasets.registry, 131](#)  
[easycv.datasets.selfsup, 109](#)  
[easycv.datasets.selfsup.data\\_sources, 109](#)  
[easycv.datasets.selfsup.data\\_sources.image\\_list, 110](#)  
[easycv.datasets.selfsup.data\\_sources.imagenet\\_features, 110](#)  
[easycv.datasets.selfsup.pipelines, 111](#)  
[easycv.datasets.selfsup.pipelines.transforms, 111](#)  
[easycv.datasets.shared, 112](#)  
[easycv.datasets.shared.base, 126](#)  
[easycv.datasets.shared.dali\\_tfrecord\\_imagenet, 127](#)  
[easycv.datasets.shared.dali\\_tfrecord\\_multi\\_view, 128](#)  
[easycv.datasets.shared.data\\_sources, 114](#)  
[easycv.datasets.shared.data\\_sources.concat, 114](#)  
[easycv.datasets.shared.data\\_sources.image\\_numpy, 114](#)  
[easycv.datasets.shared.dataset\\_wrappers, 129](#)  
[easycv.datasets.shared.multi\\_view, 129](#)  
[easycv.datasets.shared.odps\\_reader, 129](#)  
[easycv.datasets.shared.pipelines, 115](#)  
[easycv.datasets.shared.pipelines.dali\\_transformers, 115](#)  
[easycv.datasets.shared.pipelines.format, 116](#)  
[easycv.datasets.shared.pipelines.third\\_transformers\\_wrapper, 118](#)  
[easycv.datasets.shared.pipelines.transforms, 126](#)  
[easycv.datasets.shared.raw, 130](#)  
[easycv.datasets.utils, 130](#)  
[easycv.datasets.utils.tfrecord\\_util, 130](#)  
[easycv.datasets.utils.type\\_util, 131](#)  
[easycv.file, 299](#)  
[easycv.file.base, 299](#)  
[easycv.file.file\\_io, 300](#)  
[easycv.file.utils, 304](#)  
[easycv.hooks, 133](#)  
[easycv.hooks.bestckpt\\_saver\\_hook, 133](#)  
[easycv.hooks.builder, 133](#)  
[easycv.hooks.byol\\_hook, 134](#)  
[easycv.hooks.dino\\_hook, 134](#)  
[easycv.hooks.ema\\_hook, 134](#)  
[easycv.hooks.eval\\_hook, 135](#)  
[easycv.hooks.export\\_hook, 136](#)  
[easycv.hooks.extractor, 137](#)  
[easycv.hooks.optimizer\\_hook, 137](#)  
[easycv.hooks.oss\\_sync\\_hook, 137](#)  
[easycv.hooks.registry, 138](#)  
[easycv.hooks.show\\_time\\_hook, 138](#)  
[easycv.hooks.swav\\_hook, 138](#)  
[easycv.hooks.sync\\_norm\\_hook, 139](#)  
[easycv.hooks.sync\\_random\\_size\\_hook, 139](#)  
[easycv.hooks.tensorboard, 140](#)  
[easycv.hooks.wandb, 140](#)  
[easycv.hooks.yolox\\_lr\\_hook, 140](#)  
[easycv.hooks.yolox\\_mode\\_switch\\_hook, 141](#)  
[easycv.models, 191](#)  
[easycv.models.backbones, 191](#)  
[easycv.models.backbones.benchmark\\_mlp, 191](#)  
[easycv.models.backbones.bninception, 191](#)  
[easycv.models.backbones.darknet, 192](#)  
[easycv.models.backbones.genet, 193](#)  
[easycv.models.backbones.hrnet, 200](#)  
[easycv.models.backbones.inceptionv3, 204](#)  
[easycv.models.backbones.lighthrnet, 204](#)  
[easycv.models.backbones.mae\\_vit\\_transformer, 210](#)  
[easycv.models.backbones.mnasnet, 211](#)  
[easycv.models.backbones.mobilenetv2, 211](#)  
[easycv.models.backbones.network\\_blocks, 212](#)  
[easycv.models.backbones.pytorch\\_image\\_models\\_wrapper, 215](#)  
[easycv.models.backbones.resnest, 216](#)  
[easycv.models.backbones.resnet, 219](#)  
[easycv.models.backbones.resnet\\_jit, 221](#)  
[easycv.models.backbones.resnext, 224](#)  
[easycv.models.backbones.shuffle\\_transformer, 225](#)  
[easycv.models.backbones.swin\\_transformer\\_dynamic, 229](#)  
[easycv.models.backbones.vit\\_transformer\\_dynamic, 236](#)  
[easycv.models.backbones.xcit\\_transformer, 239](#)  
[easycv.models.base, 284](#)  
[easycv.models.builder, 285](#)  
[easycv.models.classification, 244](#)  
[easycv.models.classification.classification, 244](#)  
[easycv.models.classification.necks, 245](#)  
[easycv.models.detection, 248](#)  
[easycv.models.detection.utils, 248](#)  
[easycv.models.detection.utils.bboxes, 248](#)  
[easycv.models.detection.yolox, 248](#)  
[easycv.models.detection.yolox.yolo\\_head, 248](#)  
[easycv.models.detection.yolox.yolo\\_pafpn, 249](#)  
[easycv.models.detection.yolox.yolox, 250](#)  
[easycv.models.detection.yolox\\_edge, 251](#)  
[easycv.models.detection.yolox\\_edge.yolox\\_edge, 251](#)



---

easycv.models.heads, 251  
 easycv.models.heads.cls\_head, 251  
 easycv.models.heads.contrastive\_head, 252  
 easycv.models.heads.latent\_pred\_head, 253  
 easycv.models.heads.mp\_metric\_head, 253  
 easycv.models.heads.multi\_cls\_head, 254  
 easycv.models.loss, 255  
 easycv.models.loss.iou\_loss, 255  
 easycv.models.loss.mse\_loss, 255  
 easycv.models.loss.pytorch\_metric\_learning, 256  
 easycv.models.modelzoo, 285  
 easycv.models.pose, 258  
 easycv.models.pose.heads, 258  
 easycv.models.pose.heads.topdown\_heatmap\_base\_head, 259  
 easycv.models.pose.heads.topdown\_heatmap\_simple\_head, 259  
 easycv.models.pose.top\_down, 261  
 easycv.models.registry, 285  
 easycv.models.selfsup, 263  
 easycv.models.selfsup.byol, 263  
 easycv.models.selfsup.dino, 263  
 easycv.models.selfsup.mae, 265  
 easycv.models.selfsup.mixco, 266  
 easycv.models.selfsup.moby, 266  
 easycv.models.selfsup.moco, 268  
 easycv.models.selfsup.necks, 269  
 easycv.models.selfsup.simclr, 273  
 easycv.models.selfsup.swav, 274  
 easycv.models.utils, 275  
 easycv.models.utils.accuracy, 275  
 easycv.models.utils.activation, 275  
 easycv.models.utils.conv\_module, 276  
 easycv.models.utils.conv\_ws, 277  
 easycv.models.utils.dist\_utils, 278  
 easycv.models.utils.gather\_layer, 278  
 easycv.models.utils.init\_weights, 279  
 easycv.models.utils.multi\_pooling, 279  
 easycv.models.utils.norm, 280  
 easycv.models.utils.ops, 282  
 easycv.models.utils.pos\_embed, 282  
 easycv.models.utils.res\_layer, 282  
 easycv.models.utils.scale, 283  
 easycv.models.utils.sobel, 283  
 easycv.predictors, 143  
 easycv.predictors.base, 143  
 easycv.predictors.builder, 143  
 easycv.predictors.classifier, 144  
 easycv.predictors.detector, 145  
 easycv.predictors.feature\_extractor, 147  
 easycv.predictors.interface, 150  
 easycv.predictors.pose\_predictor, 152  
 easycv.runner, 305  
 easycv.runner.ev\_runner, 305  
 easycv.toolkit, 306  
 easycv.toolkit.prune, 306  
 easycv.toolkit.quantize, 306  
 easycv.toolkit.quantize.quantize\_utils, 306  
 easycv.utils, 287  
 easycv.utils.alias\_multinomial, 287  
 easycv.utils.bbox\_util, 287  
 easycv.utils.checkpoint, 288  
 easycv.utils.collect, 289  
 easycv.utils.collect\_env, 289  
 easycv.utils.config\_tools, 289  
 easycv.utils.constant, 290  
 easycv.utils.dist\_utils, 290  
 easycv.utils.eval\_utils, 291  
 easycv.utils.flops\_counter, 291  
 easycv.utils.gather, 292  
 easycv.utils.json\_utils, 292  
 easycv.utils.logger, 294  
 easycv.utils.metric\_distance, 294  
 easycv.utils.misc, 295  
 easycv.utils.preprocess\_function, 295  
 easycv.utils.profilng, 295  
 easycv.utils.py\_util, 296  
 easycv.utils.registry, 296  
 easycv.utils.test\_util, 296  
 easycv.utils.user\_config\_params\_utils, 297  
 easycv.version, 307



## Symbols

`__init__()` (`easycv.core.evaluation.auc_eval.AucEvaluator` method), 157  
`__init__()` (`easycv.core.evaluation.base_evaluator.Evaluator` method), 157  
`__init__()` (`easycv.core.evaluation.classification_eval.ClsEvaluator` method), 157  
`__init__()` (`easycv.core.evaluation.coco_evaluation.CoCoEvalWrapper` method), 160  
`__init__()` (`easycv.core.evaluation.coco_evaluation.CoCoEvalWrapper` method), 158  
`__init__()` (`easycv.core.evaluation.coco_evaluation.CoCoEvalWrapper` method), 159  
`__init__()` (`easycv.core.evaluation.coco_tools.COCOWrapper` method), 161  
`__init__()` (`easycv.core.evaluation.coco_tools.COCOWrapper` method), 161  
`__init__()` (`easycv.core.evaluation.custom_cocotools.cocoEval.COCOWrapper` method), 155  
`__init__()` (`easycv.core.evaluation.custom_cocotools.cocoEval.COCOWrapper` method), 156  
`__init__()` (`easycv.core.evaluation.faceid_pair_eval.FaceIDPairEvaluator` method), 168  
`__init__()` (`easycv.core.evaluation.metric_registry.MetricRegistry` method), 168  
`__init__()` (`easycv.core.evaluation.mse_eval.MSEEvaluator` method), 168  
`__init__()` (`easycv.core.evaluation.retrival_topk_eval.RetrialTopKEvaluator` method), 169  
`__init__()` (`easycv.core.optimizer.lars.LARS` method), 172  
`__init__()` (`easycv.core.optimizer.ranger.Ranger` method), 173  
`__init__()` (`easycv.datasets.classification.ClsDataset` method), 41  
`__init__()` (`easycv.datasets.classification.ClsOdpsDataset` method), 42  
`__init__()` (`easycv.datasets.classification.data_sources.ClsSourceCifar100` method), 45  
`__init__()` (`easycv.datasets.classification.data_sources.ClsSourceCifar100` method), 42  
`__init__()` (`easycv.datasets.classification.data_sources.ClsSourceCifar100` method), 42  
`__init__()` (`easycv.datasets.classification.data_sources.ClsSourceImageNet` method), 43  
`__init__()` (`easycv.datasets.classification.data_sources.ClsSourceImageNet` method), 43  
`__init__()` (`easycv.datasets.classification.data_sources.ClsSourceImageNet` method), 44  
`__init__()` (`easycv.datasets.classification.data_sources.cifar.ClsSourceCifar100` method), 46  
`__init__()` (`easycv.datasets.classification.data_sources.cifar.ClsSourceCifar100` method), 46  
`__init__()` (`easycv.datasets.classification.data_sources.class_list.ClsSourceImageNet` method), 47  
`__init__()` (`easycv.datasets.classification.data_sources.fashiongen_h5.FashionGenH5Dataset` method), 47  
`__init__()` (`easycv.datasets.classification.data_sources.image_list.ClsSourceImageList` method), 48  
`__init__()` (`easycv.datasets.classification.data_sources.imagenet_tfrecords.ImagenetTFRecordsDataset` method), 48  
`__init__()` (`easycv.datasets.classification.odps.ClsOdpsDataset` method), 64  
`__init__()` (`easycv.datasets.classification.pipelines.MMAutoAugment` method), 51  
`__init__()` (`easycv.datasets.classification.pipelines.MMRandAugment` method), 53  
`__init__()` (`easycv.datasets.classification.pipelines.MMRandomErasing` method), 54  
`__init__()` (`easycv.datasets.classification.pipelines.auto_augment.AutoAugment` method), 60  
`__init__()` (`easycv.datasets.classification.pipelines.auto_augment.BrightnessAugment` method), 62  
`__init__()` (`easycv.datasets.classification.pipelines.auto_augment.ColorTemperatureAugment` method), 62  
`__init__()` (`easycv.datasets.classification.pipelines.auto_augment.ContrastAugment` method), 62  
`__init__()` (`easycv.datasets.classification.pipelines.auto_augment.CutoutAugment` method), 63  
`__init__()` (`easycv.datasets.classification.pipelines.auto_augment.EqualizationAugment` method), 61  
`__init__()` (`easycv.datasets.classification.pipelines.auto_augment.InvertAugment` method), 60  
`__init__()` (`easycv.datasets.classification.pipelines.auto_augment.MMAutoAugment` method), 60

method), 56

`__init__()` (easycv.datasets.classification.pipelines.auto\_augment.MMRandAugment method), 58

`__init__()` (easycv.datasets.classification.pipelines.auto\_augment.PdG(easycv.datasets.classification.pipelines.auto\_augment method), 61

`__init__()` (easycv.datasets.classification.pipelines.auto\_augment.RoI(easycv.datasets.classification.pipelines.auto\_augment method), 60

`__init__()` (easycv.datasets.classification.pipelines.auto\_augment.SLIP(easycv.datasets.classification.pipelines.auto\_augment method), 63

`__init__()` (easycv.datasets.classification.pipelines.auto\_augment.Shift(easycv.datasets.classification.pipelines.auto\_augment method), 59

`__init__()` (easycv.datasets.classification.pipelines.auto\_augment.Solo(easycv.datasets.classification.pipelines.auto\_augment method), 61

`__init__()` (easycv.datasets.classification.pipelines.auto\_augment.SoloAdd(easycv.datasets.classification.pipelines.auto\_augment method), 61

`__init__()` (easycv.datasets.classification.pipelines.auto\_augment.Train(easycv.datasets.classification.pipelines.auto\_augment method), 59

`__init__()` (easycv.datasets.classification.pipelines.transform.MMRandAugment(easycv.datasets.classification.pipelines.transform method), 64

`__init__()` (easycv.datasets.classification.raw.ClsDataset \_\_init\_\_ method), 64

`__init__()` (easycv.datasets.detection.DetDataset \_\_init\_\_ method), 65

`__init__()` (easycv.datasets.detection.DetImagesMixDataset \_\_init\_\_ method), 66

`__init__()` (easycv.datasets.detection.data\_sources.DetSourceCoco \_\_init\_\_ method), 67

`__init__()` (easycv.datasets.detection.data\_sources.DetSourcePascal3D \_\_init\_\_ method), 69

`__init__()` (easycv.datasets.detection.data\_sources.DetSourcePascal3D \_\_init\_\_ method), 69

`__init__()` (easycv.datasets.detection.data\_sources.DetSourceVOC \_\_init\_\_ method), 70

`__init__()` (easycv.datasets.detection.data\_sources.coco.DetSourceCoco \_\_init\_\_ method), 71

`__init__()` (easycv.datasets.detection.data\_sources.pai\_format.DetSourcePai \_\_init\_\_ method), 72

`__init__()` (easycv.datasets.detection.data\_sources.raw.DetSourceRaw \_\_init\_\_ method), 73

`__init__()` (easycv.datasets.detection.data\_sources.voc.DetSourceVOC \_\_init\_\_ method), 74

`__init__()` (easycv.datasets.detection.mix.DetImagesMixDataset \_\_init\_\_ method), 94

`__init__()` (easycv.datasets.detection.pipelines.LoadAnnotation \_\_init\_\_ method), 83

`__init__()` (easycv.datasets.detection.pipelines.LoadImageFrom \_\_init\_\_ method), 82

`__init__()` (easycv.datasets.detection.pipelines.LoadMultiClassImageFrom \_\_init\_\_ method), 82

`__init__()` (easycv.datasets.detection.pipelines.MMMixUp \_\_init\_\_ method), 77

`__init__()` (easycv.datasets.detection.pipelines.MMMosaic \_\_init\_\_ method), 76

`__init__()` (easycv.datasets.detection.pipelines.MMMultiScaleFlip \_\_init\_\_ method), 76

method), 83

method), 81

method), 81

method), 81

method), 78

method), 78

method), 81

method), 79

method), 75

method), 92

method), 91

method), 91

method), 86

method), 85

method), 93

method), 91

method), 90

method), 88

method), 87

method), 90

method), 88

method), 84

method), 95

method), 97

method), 96

method), 96

method), 98

method), 99

method), 99

method), 99

\_\_init\_\_ (easycv.datasets.loader.sampler.DistributedMPSampler method), 98

\_\_init\_\_ (easycv.datasets.loader.sampler.DistributedSampler method), 98

\_\_init\_\_ (easycv.datasets.loader.sampler.GroupSampler method), 99

\_\_init\_\_ (easycv.datasets.pose.PoseTopDownDataset method), 100

\_\_init\_\_ (easycv.datasets.pose.data\_sources.PoseTopDownDataset method), 101

\_\_init\_\_ (easycv.datasets.pose.data\_sources.PoseTopDownDataset method), 101

\_\_init\_\_ (easycv.datasets.pose.data\_sources.coco.PoseTopDownDataset method), 102

\_\_init\_\_ (easycv.datasets.pose.data\_sources.top\_down.DatasetInfo method), 103

\_\_init\_\_ (easycv.datasets.pose.data\_sources.top\_down.PoseTopDownDataset method), 103

\_\_init\_\_ (easycv.datasets.pose.pipelines.PoseCollect method), 104

\_\_init\_\_ (easycv.datasets.pose.pipelines.TopDownAffine method), 105

\_\_init\_\_ (easycv.datasets.pose.pipelines.TopDownGenerative method), 105

\_\_init\_\_ (easycv.datasets.pose.pipelines.TopDownGenerative method), 105

\_\_init\_\_ (easycv.datasets.pose.pipelines.TopDownGetRandom method), 104

\_\_init\_\_ (easycv.datasets.pose.pipelines.TopDownHalfBody method), 104

\_\_init\_\_ (easycv.datasets.pose.pipelines.TopDownRandom method), 104

\_\_init\_\_ (easycv.datasets.pose.pipelines.TopDownRandom method), 106

\_\_init\_\_ (easycv.datasets.pose.pipelines.transforms.PoseCollect method), 106

\_\_init\_\_ (easycv.datasets.pose.pipelines.transforms.TopDownAffine method), 107

\_\_init\_\_ (easycv.datasets.pose.pipelines.transforms.TopDownGenerative method), 108

\_\_init\_\_ (easycv.datasets.pose.pipelines.transforms.TopDownGenerative method), 108

\_\_init\_\_ (easycv.datasets.pose.pipelines.transforms.TopDownGetRandom method), 107

\_\_init\_\_ (easycv.datasets.pose.pipelines.transforms.TopDownHalfBody method), 107

\_\_init\_\_ (easycv.datasets.pose.pipelines.transforms.TopDownRandom method), 106

\_\_init\_\_ (easycv.datasets.pose.pipelines.transforms.TopDownRandom method), 108

\_\_init\_\_ (easycv.datasets.pose.top\_down.PoseTopDownDataset method), 109

\_\_init\_\_ (easycv.datasets.selfsup.data\_sources.SSLSourceImageNetFe method), 109

\_\_init\_\_ (easycv.datasets.selfsup.data\_sources.image\_list.SSLSourceIm method), 110

\_\_init\_\_ (easycv.datasets.selfsup.data\_sources.imagenet\_feature.SSLSo method), 110

\_\_init\_\_ (easycv.datasets.selfsup.pipelines.Lighting method), 111

\_\_init\_\_ (easycv.datasets.selfsup.pipelines.RandomAppliedTrans method), 111

\_\_init\_\_ (easycv.datasets.selfsup.pipelines.Solarization method), 111

\_\_init\_\_ (easycv.datasets.selfsup.pipelines.transforms.Lighting method), 112

\_\_init\_\_ (easycv.datasets.selfsup.pipelines.transforms.MAEftAugment method), 111

\_\_init\_\_ (easycv.datasets.selfsup.pipelines.transforms.RandomAppliedD method), 111

\_\_init\_\_ (easycv.datasets.selfsup.pipelines.transforms.Solarization method), 112

\_\_init\_\_ (easycv.datasets.shared.BaseDataset method), 113

\_\_init\_\_ (easycv.datasets.shared.ConcatDataset method), 112

\_\_init\_\_ (easycv.datasets.shared.MultiViewDataset method), 113

\_\_init\_\_ (easycv.datasets.shared.OdpsReader method), 112

\_\_init\_\_ (easycv.datasets.shared.RawDataset method), 113

\_\_init\_\_ (easycv.datasets.shared.RepeatDataset method), 112

\_\_init\_\_ (easycv.datasets.shared.base.BaseDataset method), 126

\_\_init\_\_ (easycv.datasets.shared.dali\_tfrecord\_imagenet.DaliImageNet method), 127

\_\_init\_\_ (easycv.datasets.shared.dali\_tfrecord\_imagenet.DaliLoaderW method), 127

\_\_init\_\_ (easycv.datasets.shared.dali\_tfrecord\_imagenet.ImageNetTFR method), 127

\_\_init\_\_ (easycv.datasets.shared.dali\_tfrecord\_multi\_view.DaliLoader method), 128

\_\_init\_\_ (easycv.datasets.shared.dali\_tfrecord\_multi\_view.DaliTFReco method), 128

\_\_init\_\_ (easycv.datasets.shared.dali\_tfrecord\_multi\_view.DaliTFReco method), 128

\_\_init\_\_ (easycv.datasets.shared.data\_sources.ImageNpy method), 114

\_\_init\_\_ (easycv.datasets.shared.data\_sources.SourceConcat method), 114

\_\_init\_\_ (easycv.datasets.shared.data\_sources.concat.SourceConcat method), 114

\_\_init\_\_ (easycv.datasets.shared.data\_sources.image\_npy.ImageNpy method), 114

|                                                                                                                              |                                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <code>method)</code> , 114                                                                                                   | <code>method)</code> , 137                                                                                  |
| <code>__init__()</code> ( <code>easycv.datasets.shared.dataset_wrappers.ConcatDataset</code> method), 129                    | <code>__init__()</code> ( <code>easycv.hooks.optimizer_hook.OptimizerHook</code> method), 137               |
| <code>__init__()</code> ( <code>easycv.datasets.shared.dataset_wrappers.RepeatDataset</code> method), 129                    | <code>__init__()</code> ( <code>easycv.hooks.oss_sync_hook.OSSSyncHook</code> method), 137                  |
| <code>__init__()</code> ( <code>easycv.datasets.shared.multi_view.MultiViewDataset</code> method), 129                       | <code>__init__()</code> ( <code>easycv.hooks.show_time_hook.TIMEHook</code> method), 138                    |
| <code>__init__()</code> ( <code>easycv.datasets.shared.odps_reader.OdpsReader</code> method), 129                            | <code>__init__()</code> ( <code>easycv.hooks.swav_hook.SWAVHook</code> method), 138                         |
| <code>__init__()</code> ( <code>easycv.datasets.shared.pipelines.dali_transforms.DaliCropTwist</code> method), 115           | <code>__init__()</code> ( <code>easycv.hooks.sync_norm_hook.SyncNormHook</code> method), 139                |
| <code>__init__()</code> ( <code>easycv.datasets.shared.pipelines.dali_transforms.DaliCropMirrorNormalize</code> method), 116 | <code>__init__()</code> ( <code>easycv.hooks.sync_random_size_hook.SyncRandomSizeHook</code> method), 139   |
| <code>__init__()</code> ( <code>easycv.datasets.shared.pipelines.dali_transforms.DaliImageDecoder</code> method), 115        | <code>__init__()</code> ( <code>easycv.hooks.yolox_lr_hook.YOLOXLRUpdaterHook</code> method), 140           |
| <code>__init__()</code> ( <code>easycv.datasets.shared.pipelines.dali_transforms.DaliRawImageHook</code> method), 116        | <code>__init__()</code> ( <code>easycv.hooks.yolox_mode_switch_hook.YOLOXModeSwitchHook</code> method), 141 |
| <code>__init__()</code> ( <code>easycv.datasets.shared.pipelines.dali_transforms.DaliRawImageResizeCrop</code> method), 115  | <code>__init__()</code> ( <code>easycv.models.backbones.benchmark_mlp.BenchMarkMLP</code> method), 191      |
| <code>__init__()</code> ( <code>easycv.datasets.shared.pipelines.dali_transforms.DaliResizeCrop</code> method), 115          | <code>__init__()</code> ( <code>easycv.models.backbones.bninception.BNInception</code> method), 191         |
| <code>__init__()</code> ( <code>easycv.datasets.shared.pipelines.format.Collector</code> method), 117                        | <code>__init__()</code> ( <code>easycv.models.backbones.darknet.CSPDarknet</code> method), 192              |
| <code>__init__()</code> ( <code>easycv.datasets.shared.pipelines.format.ImageToTensor</code> method), 116                    | <code>__init__()</code> ( <code>easycv.models.backbones.darknet.Darknet</code> method), 192                 |
| <code>__init__()</code> ( <code>easycv.datasets.shared.pipelines.transforms.Compose</code> method), 126                      | <code>__init__()</code> ( <code>easycv.models.backbones.genet.AdaptiveAvgPool</code> method), 193           |
| <code>__init__()</code> ( <code>easycv.datasets.shared.raw.RawDataset</code> method), 130                                    | <code>__init__()</code> ( <code>easycv.models.backbones.genet.BN</code> method), 194                        |
| <code>__init__()</code> ( <code>easycv.file.file_io.BinaryOSSFile</code> method), 304                                        | <code>__init__()</code> ( <code>easycv.models.backbones.genet.ConvDW</code> method), 194                    |
| <code>__init__()</code> ( <code>easycv.file.file_io.IO</code> method), 300                                                   | <code>__init__()</code> ( <code>easycv.models.backbones.genet.ConvKX</code> method), 194                    |
| <code>__init__()</code> ( <code>easycv.file.file_io.NullContextWrapper</code> method), 304                                   | <code>__init__()</code> ( <code>easycv.models.backbones.genet.Flatten</code> method), 195                   |
| <code>__init__()</code> ( <code>easycv.file.file_io.OSSFile</code> method), 304                                              | <code>__init__()</code> ( <code>easycv.models.backbones.genet.Linear</code> method), 195                    |
| <code>__init__()</code> ( <code>easycv.hooks.best_ckpt_saver_hook.BestCkptSaverHook</code> method), 133                      | <code>__init__()</code> ( <code>easycv.models.backbones.genet.Linear</code> method), 195                    |
| <code>__init__()</code> ( <code>easycv.hooks.byol_hook.BYOLHook</code> method), 134                                          | <code>__init__()</code> ( <code>easycv.models.backbones.genet.MaxPool</code> method), 196                   |
| <code>__init__()</code> ( <code>easycv.hooks.dino_hook.DINOHook</code> method), 134                                          | <code>__init__()</code> ( <code>easycv.models.backbones.genet.MultiSumBlock</code> method), 196             |
| <code>__init__()</code> ( <code>easycv.hooks.ema_hook.EMAHook</code> method), 135                                            | <code>__init__()</code> ( <code>easycv.models.backbones.genet.PlainNet</code> method), 200                  |
| <code>__init__()</code> ( <code>easycv.hooks.ema_hook.ModelEMA</code> method), 134                                           | <code>__init__()</code> ( <code>easycv.models.backbones.genet.PlainNetBasicBlockClass</code> method), 193   |
| <code>__init__()</code> ( <code>easycv.hooks.eval_hook.DistEvalHook</code> method), 136                                      | <code>__init__()</code> ( <code>easycv.models.backbones.genet.RELU</code> method), 196                      |
| <code>__init__()</code> ( <code>easycv.hooks.eval_hook.EvalHook</code> method), 135                                          | <code>__init__()</code> ( <code>easycv.models.backbones.genet.ResBlock</code> method), 197                  |
| <code>__init__()</code> ( <code>easycv.hooks.export_hook.ExportHook</code> method), 136                                      | <code>__init__()</code> ( <code>easycv.models.backbones.genet.Sequential</code> method), 197                |
| <code>__init__()</code> ( <code>easycv.hooks.extractor.Extractor</code> method), 137                                         | <code>__init__()</code> ( <code>easycv.models.backbones.genet.SuperResK1DW</code> method), 199              |
| <code>__init__()</code> ( <code>easycv.hooks.optimizer_hook.AMPFP16OptimizerHook</code> method), 137                         | <code>__init__()</code> ( <code>easycv.models.backbones.genet.SuperResK1DWK1</code> method), 199            |



`method`), 199  
`__init__` () (`easycv.models.backbones.genet.SuperResK1KX` `__init__` () (`easycv.models.backbones.pytorch_image_models_wrapper.Py`  
`method`), 198  
`__init__` () (`easycv.models.backbones.genet.SuperResK1KXK1K1` `__init__` () (`easycv.models.backbones.resnest.Bottleneck`  
`method`), 198  
`__init__` () (`easycv.models.backbones.genet.SuperResKXXKX` `__init__` () (`easycv.models.backbones.resnest.DropBlock2D`  
`method`), 198  
`__init__` () (`easycv.models.backbones.hrnet.Bottleneck` `__init__` () (`easycv.models.backbones.resnest.GlobalAvgPool2d`  
`method`), 201  
`__init__` () (`easycv.models.backbones.hrnet.HRModule` `__init__` () (`easycv.models.backbones.resnest.ResNeSt`  
`method`), 201  
`__init__` () (`easycv.models.backbones.hrnet.HRNet` `__init__` () (`easycv.models.backbones.resnest.SplAtConv2d`  
`method`), 203  
`__init__` () (`easycv.models.backbones.inceptionv3.InceptionV3` `__init__` () (`easycv.models.backbones.resnest.rSoftMax`  
`method`), 204  
`__init__` () (`easycv.models.backbones.lighthrnet.ConditionalGrouping` `__init__` () (`easycv.models.backbones.resnet.BasicBlock`  
`method`), 206  
`__init__` () (`easycv.models.backbones.lighthrnet.CrossResolutionWeighting` `__init__` () (`easycv.models.backbones.resnet.Bottleneck`  
`method`), 205  
`__init__` () (`easycv.models.backbones.lighthrnet.IterativeHead` `__init__` () (`easycv.models.backbones.resnet.ResNet`  
`method`), 207  
`__init__` () (`easycv.models.backbones.lighthrnet.LiteHRModule` `__init__` () (`easycv.models.backbones.resnet_jit.BasicBlock`  
`method`), 208  
`__init__` () (`easycv.models.backbones.lighthrnet.LiteHRNet` `__init__` () (`easycv.models.backbones.resnet_jit.Bottleneck`  
`method`), 209  
`__init__` () (`easycv.models.backbones.lighthrnet.ShuffleUnit` `__init__` () (`easycv.models.backbones.resnet_jit.ResNetJIT`  
`method`), 208  
`__init__` () (`easycv.models.backbones.lighthrnet.SpatialWeighting` `__init__` () (`easycv.models.backbones.resnext.Bottleneck`  
`method`), 205  
`__init__` () (`easycv.models.backbones.lighthrnet.Stem` `__init__` () (`easycv.models.backbones.resnext.ResNeXt`  
`method`), 207  
`__init__` () (`easycv.models.backbones.mae_vit_transformer.MaskedAutoencoders` `__init__` () (`easycv.models.backbones.shuffle_transformer.Attention`  
`method`), 210  
`__init__` () (`easycv.models.backbones.mnasnet.MNASNet` `__init__` () (`easycv.models.backbones.shuffle_transformer.Block`  
`method`), 211  
`__init__` () (`easycv.models.backbones.mobilenetv2.MobileNetV2` `__init__` () (`easycv.models.backbones.shuffle_transformer.Mlp`  
`method`), 211  
`__init__` () (`easycv.models.backbones.network_blocks.BaseConv` `__init__` () (`easycv.models.backbones.shuffle_transformer.PatchEmbedding`  
`method`), 212  
`__init__` () (`easycv.models.backbones.network_blocks.Bottleneck` `__init__` () (`easycv.models.backbones.shuffle_transformer.PatchMerging`  
`method`), 213  
`__init__` () (`easycv.models.backbones.network_blocks.CSPLayer` `__init__` () (`easycv.models.backbones.shuffle_transformer.ShuffleTransformer`  
`method`), 214  
`__init__` () (`easycv.models.backbones.network_blocks.DWConv` `__init__` () (`easycv.models.backbones.shuffle_transformer.StageModule`  
`method`), 213  
`__init__` () (`easycv.models.backbones.network_blocks.Focus` `__init__` () (`easycv.models.backbones.swin_transformer_dynamic.BasicLayer`  
`method`), 215  
`__init__` () (`easycv.models.backbones.network_blocks.HSiLU` `__init__` () (`easycv.models.backbones.swin_transformer_dynamic.Mlp`  
`method`), 212  
`__init__` () (`easycv.models.backbones.network_blocks.ResLayer` `__init__` () (`easycv.models.backbones.swin_transformer_dynamic.PatchEmbedding`  
`method`), 213  
`__init__` () (`easycv.models.backbones.network_blocks.SPPBottleneck` `__init__` () (`easycv.models.backbones.swin_transformer_dynamic.PatchMerging`  
`method`), 214  
`__init__` () (`easycv.models.backbones.network_blocks.SiLU` `__init__` () (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformer`  
`method`), 212

method), 235

`__init__()` (`easycv.models.backbones.swin_transformer_dynamic_vision_transformer` method), 231

`__init__()` (`easycv.models.backbones.swin_transformer_dynamic_vision_transformer` method), 230

`__init__()` (`easycv.models.backbones.vit_transformer_dynamic_vision_transformer` method), 236

`__init__()` (`easycv.models.backbones.vit_transformer_dynamic_vision_transformer` method), 237

`__init__()` (`easycv.models.backbones.vit_transformer_dynamic_vision_transformer` method), 236

`__init__()` (`easycv.models.backbones.vit_transformer_dynamic_vision_transformer` method), 236

`__init__()` (`easycv.models.backbones.vit_transformer_dynamic_vision_transformer` method), 237

`__init__()` (`easycv.models.backbones.vit_transformer_dynamic_vision_transformer` method), 238

`__init__()` (`easycv.models.backbones.xcit_transformer.ClassificationHead` method), 240

`__init__()` (`easycv.models.backbones.xcit_transformer.ClassificationHead` method), 241

`__init__()` (`easycv.models.backbones.xcit_transformer.Conv2d` method), 239

`__init__()` (`easycv.models.backbones.xcit_transformer.LPI` method), 240

`__init__()` (`easycv.models.backbones.xcit_transformer.PositionalEncoding` method), 239

`__init__()` (`easycv.models.backbones.xcit_transformer.XCA` method), 241

`__init__()` (`easycv.models.backbones.xcit_transformer.XCABlock` method), 242

`__init__()` (`easycv.models.backbones.xcit_transformer.XCi` method), 242

`__init__()` (`easycv.models.classification.classification.Classification` method), 244

`__init__()` (`easycv.models.classification.necks.FaceIDNeck` method), 246

`__init__()` (`easycv.models.classification.necks.HRFuseScale` method), 247

`__init__()` (`easycv.models.classification.necks.LinearNeck` method), 245

`__init__()` (`easycv.models.classification.necks.MultiLinearNeck` method), 246

`__init__()` (`easycv.models.classification.necks.ReutralNeck` method), 245

`__init__()` (`easycv.models.detection.yolox.yolo_head.YOLOXHead` method), 248

`__init__()` (`easycv.models.detection.yolox.yolo_pafpn.YOLOXPAFPN` method), 249

`__init__()` (`easycv.models.detection.yolox.yolox.YOLOX` method), 250

`__init__()` (`easycv.models.detection.yolox_edge.yolox_edge.YOLOXEdge` method), 251

`__init__()` (`easycv.models.heads.cls_head.ClsHead` method), 251

`__init__()` (`easycv.models.heads.cls_head.ClsHead` method), 252

`__init__()` (`easycv.models.heads.cls_head.ClsHead` method), 252

`__init__()` (`easycv.models.heads.latent_pred_head.LatentClsHead` method), 253

`__init__()` (`easycv.models.heads.latent_pred_head.LatentPredictHead` method), 253

`__init__()` (`easycv.models.heads.mp_metric_head.MpMetricHead` method), 254

`__init__()` (`easycv.models.heads.multi_cls_head.MultiClsHead` method), 254

`__init__()` (`easycv.models.loss.iou_loss.IOULoss` method), 255

`__init__()` (`easycv.models.loss.mse_loss.JointsMSELoss` method), 255

`__init__()` (`easycv.models.loss.pytorch_metric_learning.AMSoftmaxLoss` method), 257

`__init__()` (`easycv.models.loss.pytorch_metric_learning.CrossEntropyLoss` method), 256

`__init__()` (`easycv.models.loss.pytorch_metric_learning.DistributeMSELoss` method), 256

`__init__()` (`easycv.models.loss.pytorch_metric_learning.FocalLoss2d` method), 256

`__init__()` (`easycv.models.loss.pytorch_metric_learning.ModelParallelA` method), 258

`__init__()` (`easycv.models.loss.pytorch_metric_learning.ModelParallelSc` method), 257

`__init__()` (`easycv.models.loss.pytorch_metric_learning.SoftTargetCross` method), 258

`__init__()` (`easycv.models.pose.heads.topdown_heatmap_simple_head.TopDown` method), 260

`__init__()` (`easycv.models.pose.top_down.TopDown` method), 261

`__init__()` (`easycv.models.selfsup.byol.BYOL` method), 263

`__init__()` (`easycv.models.selfsup.dino.DINO` method), 264

`__init__()` (`easycv.models.selfsup.dino.DINOLoss` method), 264

`__init__()` (`easycv.models.selfsup.dino.MultiCropWrapper` method), 263

`__init__()` (`easycv.models.selfsup.mae.MAE` method), 265

`__init__()` (`easycv.models.selfsup.mixco.MIXCO` method), 266

`__init__()` (`easycv.models.selfsup.moby.MoBY` method), 266

`__init__()` (`easycv.models.selfsup.moco.MOCO` method), 268

`__init__()` (`easycv.models.selfsup.necks.DINONeck` method), 269

`__init__()` (`easycv.models.selfsup.necks.MAENeck` method), 269



`method`), 272  
`__init__()` (`easycv.models.selfsup.necks.MoBYMLP` `method`), 269  
`__init__()` (`easycv.models.selfsup.necks.NonLinearNeckSimCLR` `method`), 271  
`__init__()` (`easycv.models.selfsup.necks.NonLinearNeckSwAV` `method`), 269  
`__init__()` (`easycv.models.selfsup.necks.NonLinearNeckV0` `method`), 270  
`__init__()` (`easycv.models.selfsup.necks.NonLinearNeckV1` `method`), 270  
`__init__()` (`easycv.models.selfsup.necks.NonLinearNeckV2` `method`), 271  
`__init__()` (`easycv.models.selfsup.necks.RelativeLocNeck` `method`), 272  
`__init__()` (`easycv.models.selfsup.simclr.SimCLR` `method`), 273  
`__init__()` (`easycv.models.selfsup.swav.MultiPrototypes` `method`), 274  
`__init__()` (`easycv.models.selfsup.swav.SWAV` `method`), 274  
`__init__()` (`easycv.models.utils.accuracy.Accuracy` `method`), 275  
`__init__()` (`easycv.models.utils.activation.FReLU` `method`), 275  
`__init__()` (`easycv.models.utils.conv_module.ConvModule` `method`), 276  
`__init__()` (`easycv.models.utils.conv_ws.ConvWS2d` `method`), 277  
`__init__()` (`easycv.models.utils.dist_utils.DistributedLossWrapper` `method`), 278  
`__init__()` (`easycv.models.utils.dist_utils.DistributedMinerWrapper` `method`), 278  
`__init__()` (`easycv.models.utils.multi_pooling.GeMPooling` `method`), 279  
`__init__()` (`easycv.models.utils.multi_pooling.MultiAvgPooling` `method`), 280  
`__init__()` (`easycv.models.utils.multi_pooling.MultiPooling` `method`), 280  
`__init__()` (`easycv.models.utils.norm.IBN` `method`), 281  
`__init__()` (`easycv.models.utils.norm.SyncIBN` `method`), 280  
`__init__()` (`easycv.models.utils.res_layer.ResLayer` `method`), 283  
`__init__()` (`easycv.models.utils.scale.Scale` `method`), 283  
`__init__()` (`easycv.models.utils.sobel.Sobel` `method`), 283  
`__init__()` (`easycv.predictors.base.Predictor` `method`), 143  
`__init__()` (`easycv.predictors.classifier.TorchClassifier` `method`), 144  
`__init__()` (`easycv.predictors.detector.TorchFaceDetector` `method`), 145  
`__init__()` (`easycv.predictors.detector.TorchYoloXClassifierPredictor` `method`), 146  
`__init__()` (`easycv.predictors.detector.TorchYoloXPredictor` `method`), 145  
`__init__()` (`easycv.predictors.feature_extractor.TorchFaceAttrExtractor` `method`), 149  
`__init__()` (`easycv.predictors.feature_extractor.TorchFaceFeatureExtractor` `method`), 147  
`__init__()` (`easycv.predictors.feature_extractor.TorchFeatureExtractor` `method`), 147  
`__init__()` (`easycv.predictors.feature_extractor.TorchMultiFaceFeatureExtractor` `method`), 148  
`__init__()` (`easycv.predictors.interface.PredictorInterface` `method`), 150  
`__init__()` (`easycv.predictors.interface.PredictorInterfaceV2` `method`), 151  
`__init__()` (`easycv.predictors.pose_predictor.LoadImage` `method`), 152  
`__init__()` (`easycv.predictors.pose_predictor.OutputHook` `method`), 152  
`__init__()` (`easycv.predictors.pose_predictor.TorchPoseTopDownPredictor` `method`), 152  
`__init__()` (`easycv.predictors.pose_predictor.TorchPoseTopDownPredictorV2` `method`), 153  
`__init__()` (`easycv.runner.ev_runner.EVRunner` `method`), 305  
`__init__()` (`easycv.utils.alias_multinomial.AliasMethod` `method`), 287  
`__init__()` (`easycv.utils.registry.Registry` `method`), 296  
`abspath()` (`easycv.file.base.IOBase` `method`), 299  
`abspath()` (`easycv.file.base.IOLocal` `method`), 300  
`abspath()` (`easycv.file.file_io.IO` `method`), 304  
`access_oss()` (`easycv.file.file_io.IO` `method`), 300  
`accumulate()` (`easycv.core.evaluation.custom_cocotools.cocoeval.COCOEval` `method`), 155  
`Accuracy` (class in `easycv.models.utils.accuracy`), 275  
`accuracy()` (in module `easycv.models.utils.accuracy`), 275  
`AdaptiveAvgPool` (class in `easycv.models.backbones.genet`), 193  
`add_batch_counter_hook_function()` (in module `easycv.utils.flops_counter`), 292  
`add_batch_counter_variables_or_reset()` (in module `easycv.utils.flops_counter`), 292  
`add_flops_counter_hook_function()` (in module `easycv.utils.flops_counter`), 292  
`add_flops_counter_variable_or_reset()` (in module `easycv.utils.flops_counter`), 292  
`add_flops_counting_methods()` (in module `easycv.utils.flops_counter`), 291

`add_flops_mask()` (in module `easycv.utils.flops_counter`), 291  
`add_flops_mask_variable_or_reset()` (in module `easycv.utils.flops_counter`), 292  
`add_single_detected_image_info()` (`easycv.core.evaluation.coco_evaluation.CocoDetectionEvaluator` method), 159  
`add_single_detected_image_info()` (`easycv.core.evaluation.coco_evaluation.CocoMaskEvaluator` method), 160  
`add_single_ground_truth_image_info()` (`easycv.core.evaluation.coco_evaluation.CocoDetectionEvaluator` method), 158  
`add_single_ground_truth_image_info()` (`easycv.core.evaluation.coco_evaluation.CocoMaskEvaluator` method), 159  
`add_visualization_info()` (`easycv.hooks.eval_hook.EvalHook` method), 135  
`affine_transform()` (in module `easycv.core.post_processing`), 173  
`affine_transform()` (in module `easycv.core.post_processing.pose_transforms`), 180  
`after_run()` (`easycv.hooks.export_hook.ExportHook` method), 136  
`after_run()` (`easycv.hooks.oss_sync_hook.OSSSyncHook` method), 138  
`after_train_epoch()` (`easycv.hooks.bestckpt_saver_hook.BestCkptSaverHook` method), 133  
`after_train_epoch()` (`easycv.hooks.eval_hook.DistEvalHook` method), 136  
`after_train_epoch()` (`easycv.hooks.eval_hook.EvalHook` method), 135  
`after_train_epoch()` (`easycv.hooks.export_hook.ExportHook` method), 136  
`after_train_epoch()` (`easycv.hooks.oss_sync_hook.OSSSyncHook` method), 138  
`after_train_epoch()` (`easycv.hooks.swav_hook.SWAVHook` method), 138  
`after_train_epoch()` (`easycv.hooks.sync_norm_hook.SyncNormHook` method), 139  
`after_train_iter()` (`easycv.hooks.dino_hook.DINOHook` method), 134  
`after_train_iter()` (`easycv.hooks.ema_hook.EMAHook` method), 135  
`after_train_iter()` (`easycv.hooks.export_hook.ExportHook` method), 136  
`after_train_iter()` (`easycv.hooks.optimizer_hook.AMPFP16OptimizerHook` method), 137  
`after_train_iter()` (`easycv.hooks.optimizer_hook.OptimizerHook` method), 137  
`after_train_iter()` (`easycv.hooks.oss_sync_hook.OSSSyncHook` method), 138  
`after_train_iter()` (`easycv.hooks.show_time_hook.TIMEHook` method), 138  
`after_train_iter()` (`easycv.hooks.sync_random_size_hook.SyncRandomSizeHook` method), 139  
`after_train_iter()` (`easycv.hooks.tensorboard.TensorboardLoggerHook` method), 140  
`after_train_iter()` (`easycv.hooks.wandb.WandbLoggerHookV2` method), 140  
`AliasMethod` (class in `easycv.utils.alias_multinomial`), 287  
`all_gather()` (in module `easycv.models.utils.dist_utils`), 278  
`all_gather_embeddings_labels()` (in module `easycv.models.utils.dist_utils`), 278  
`all_reduce_dict()` (in module `easycv.utils.dist_utils`), 290  
`AMPFP16OptimizerHook` (class in `easycv.hooks.optimizer_hook`), 137  
`AMSoftmaxLoss` (class in `easycv.models.loss.pytorch_metric_learning`), 257  
`Analyze()` (`easycv.core.evaluation.coco_tools.COCOEvalWrapper` method), 163  
`analyze()` (`easycv.core.evaluation.custom_cocotools.cocoeval.COCOEval` method), 156  
`ap_per_class()` (in module `easycv.core.evaluation.ap`), 156  
`arch_settings` (`easycv.models.backbones.resnest.ResNeSt` attribute), 218  
`arch_settings` (`easycv.models.backbones.resnet.ResNet` attribute), 220  
`arch_settings` (`easycv.models.backbones.resnet_jit.ResNetJIT` attribute), 223  
`arch_settings` (`easycv.models.backbones.resnext.ResNeXt` attribute), 225  
`arch_zoo` (`easycv.models.backbones.hrnet.HRNet` attribute), 203  
`Attention` (class in `easycv.models.backbones.shuffle_transformer`), 226  
`Attention` (class in `easycv.models.backbones.vit_transformer_dynamic`), 236  
`AucEvaluator` (class in `easycv.core.evaluation.auc_eval`), 157  
`aug_test()` (`easycv.models.classification.classification.Classification` method), 244  
`authorize()` (`easycv.file_io.IO` method), 304  
`AutoAugment` (class in `easycv.models.backbones.vit_transformer_dynamic`), 236

`easycv.datasets.shared.pipelines.third_transforms.before_run()` (`easycv.hooks.dino_hook.DINOHook` method), 134  
 118  
**AutoContrast** (class in `before_run()` (`easycv.hooks.ema_hook.EMAHook` method), 135  
`easycv.datasets.classification.pipelines.auto_augment`), method), 135  
 60  
**B**  
`before_run()` (`easycv.hooks.eval_hook.DistEvalHook` method), 136  
`before_run()` (`easycv.hooks.eval_hook.EvalHook` method), 136  
**b64\_decode()** (`easycv.datasets.shared.odps_reader.OdpsReader` method), 135  
 method), 130  
**b64\_decode()** (`easycv.datasets.shared.OdpsReader` method), 137  
 method), 113  
**backward()** (`easycv.models.utils.gather_layer.GatherLayer` method), 137  
 static method), 279  
**barrier()** (in module `easycv.utils.dist_utils`), 290  
**BaseConv** (class in `easycv.models.backbones.network_block.before_train_epoch()`  
 212 (`easycv.hooks.dino_hook.DINOHook` method), 134  
**BaseDataset** (class in `easycv.datasets.shared`), 113  
**BaseDataset** (class in `easycv.datasets.shared.base`), 126 `before_train_epoch()`  
**BaseModel** (class in `easycv.models.base`), 284 (`easycv.hooks.ema_hook.EMAHook` method), 135  
**BasicBlock** (class in `easycv.models.backbones.resnet`), `before_train_epoch()`  
 219 (`easycv.hooks.swav_hook.SWAVHook` method), 138  
**BasicBlock** (class in `easycv.models.backbones.resnet_jit`), 138  
 221  
**BasicLayer** (class in `easycv.models.backbones.swin_transformer.before_train_epoch()`  
 232 (`easycv.hooks.sync_norm_hook.SyncNormHook` method), 139  
**batch()** (`easycv.predictors.classifier.TorchClassifier` method), 144  
`before_train_epoch()`  
**batch()** (`easycv.predictors.detector.TorchFaceDetector` method), 145 (`easycv.hooks.yolox_mode_switch_hook.YOLOXModeSwitchHook` method), 141  
**batch()** (`easycv.predictors.feature_extractor.TorchFaceAttention.before_train_iter()`  
 method), 150 (`easycv.hooks.byol_hook.BYOLHook` method), 134  
**batch()** (`easycv.predictors.feature_extractor.TorchFaceFeatureExtractor` method), 148  
`before_train_iter()`  
**batch()** (`easycv.predictors.feature_extractor.TorchFeatureExtractor` method), 147 (`easycv.hooks.dino_hook.DINOHook` method), 134  
**batch()** (`easycv.predictors.feature_extractor.TorchMultiFaceFeatureExtractor` method), 149  
`before_train_iter()`  
**batch\_counter\_hook()** (in module `before_train_iter()`  
`easycv.utils.flops_counter`), 292 (`easycv.hooks.show_time_hook.TIMEHook` method), 138  
**batch\_size** (`easycv.datasets.loader.build_loader.InfiniteDataLoader` method), 295  
 attribute), 98  
**BenchmarkMLP** (class in `benchmark_torch_function()` (in module  
`easycv.models.backbones.benchmark_mlp`), 191  
**batched\_cxxywh2xyxy\_with\_shape()** (in module `easycv.models.backbones.benchmark_mlp`), 191  
`easycv.utils.bbox_util`), 287  
**batched\_xyxy2cxxywh\_with\_shape()** (in module `BestCkptSaverHook` (class in  
`easycv.utils.bbox_util`), 287 `easycv.hooks.best_ckpt_saver_hook`), 133  
**bbox\_flip()** (`easycv.datasets.detection.pipelines.mm_transforms.conv_ws.ConvWS2d` attribute),  
 method), 90 277  
**bbox\_flip()** (`easycv.datasets.detection.pipelines.MMRandomFlip` method), 81  
**BinaryOSSFile** (class in `easycv.file.file_io`), 304  
**Block** (class in `easycv.models.backbones.shuffle_transformer`), 226  
**Block** (class in `easycv.models.backbones.vit_transformer_dynamic`), 237  
**before\_run()** (`easycv.hooks.best_ckpt_saver_hook.BestCkptSaverHook` method), 133  
`before_run()` (`easycv.models.backbones.hrnet.HRNet` attribute), 202

- BN (class in *easycv.models.backbones.genet*), 194
- bn\_flops\_counter\_hook() (in module *easycv.utils.flops\_counter*), 292
- BNInception (class in *easycv.models.backbones.bninception*), 191
- bninceptionPre() (in module *easycv.utils.preprocess\_function*), 295
- Bottleneck (class in *easycv.models.backbones.hrnet*), 200
- Bottleneck (class in *easycv.models.backbones.network\_block*), 213
- Bottleneck (class in *easycv.models.backbones.resnet*), 217
- Bottleneck (class in *easycv.models.backbones.resnet*), 219
- Bottleneck (class in *easycv.models.backbones.resnet\_jit*), 222
- Bottleneck (class in *easycv.models.backbones.resnext*), 224
- bound\_limits() (in module *easycv.utils.bbox\_util*), 287
- bound\_limits\_for\_list() (in module *easycv.utils.bbox\_util*), 287
- box\_candidates() (in module *easycv.utils.bbox\_util*), 288
- box\_iou() (in module *easycv.utils.bbox\_util*), 288
- Brightness (class in *easycv.datasets.classification.pipelines.auto\_augment*), 62
- build() (in module *easycv.models.builder*), 285
- build\_backbone() (in module *easycv.models.builder*), 285
- build\_conv\_layer() (in module *easycv.models.utils.conv\_module*), 276
- build\_dali\_dataset() (in module *easycv.datasets.builder*), 131
- build\_data\_loader() (in module *easycv.datasets.loader*), 96
- build\_data\_loader() (in module *easycv.datasets.loader.build\_loader*), 97
- build\_dataset() (in module *easycv.datasets.builder*), 131
- build\_datasource() (in module *easycv.datasets.builder*), 131
- build\_evaluator() (in module *easycv.core.evaluation.builder*), 157
- build\_from\_cfg() (in module *easycv.utils.registry*), 296
- build\_head() (in module *easycv.models.builder*), 285
- build\_hook() (in module *easycv.hooks.builder*), 133
- build\_loss() (in module *easycv.models.builder*), 285
- build\_memory() (in module *easycv.models.builder*), 285
- build\_model() (in module *easycv.models.builder*), 285
- build\_neck() (in module *easycv.models.builder*), 285
- build\_norm\_layer() (in module *easycv.models.utils.norm*), 281
- build\_optimizer() (in module *easycv.apis.train*), 39
- build\_predictor() (in module *easycv.predictors.builder*), 143
- build\_yolo\_optimizer() (in module *easycv.apis.train\_misc*), 39
- BYOL (class in *easycv.models.selfsup.byol*), 263
- BYOLHook (class in *easycv.hooks.byol\_hook*), 134
- calculate\_accuracy() (in module *easycv.core.evaluation.faceid\_pair\_eval*), 167
- calculate\_roc() (in module *easycv.core.evaluation.faceid\_pair\_eval*), 167
- calculate\_this\_label\_list() (in module *easycv.datasets.loader.sampler.DistributedMPSampler* method), 98
- calculate\_val() (in module *easycv.core.evaluation.faceid\_pair\_eval*), 167
- calculate\_val\_far() (in module *easycv.core.evaluation.faceid\_pair\_eval*), 167
- call() (in module *easycv.toolkit.quantize.quantize\_utils*), 306
- CenterCrop (class in *easycv.datasets.shared.pipelines.third\_transforms\_wrapper*), 118
- centralized\_gradient() (in module *easycv.core.optimizer.ranger*), 173
- channel\_shuffle() (in module *easycv.models.backbones.lightrnet*), 204
- channels (in *easycv.core.standard\_fields.TfExampleFields* attribute), 187, 188
- char\_dict (in *easycv.core.standard\_fields.InputDataFields* attribute), 186
- check\_base\_cfg\_path() (in module *easycv.utils.config\_tools*), 289
- check\_value\_type() (in module *easycv.utils.user\_config\_params\_utils*), 297
- ClassAttention (class in *easycv.models.backbones.xcit\_transformer*), 240
- ClassAttentionBlock (class in *easycv.models.backbones.xcit\_transformer*), 240
- CLASSES (in *easycv.datasets.classification.data\_sources.cifar.ClsSourceCifar10* attribute), 46
- CLASSES (in *easycv.datasets.classification.data\_sources.cifar.ClsSourceCifar100* attribute), 46
- CLASSES (in *easycv.datasets.classification.data\_sources.ClsSourceCifar10* attribute), 42



**CLASSES** (*easycv.datasets.classification.data\_sources.ClsSourceCifar10* (class in *easycv.datasets.classification.data\_sources*), attribute), 42  
**CLASSES** (*easycv.datasets.classification.data\_sources.ClsSourceCUB* (class in *easycv.datasets.classification.data\_sources*), attribute), 44  
**Classification** (class in *easycv.datasets.classification.data\_sources.imagenet\_tfrecord*), *easycv.models.classification.classification*), 244  
**clean\_up()** (in module *easycv.utils.test\_util*), 296  
**clear()** (*easycv.core.evaluation.coco\_evaluation.CocoDetectionEvaluator* method), 158  
**clear()** (*easycv.core.evaluation.coco\_evaluation.CocoMaskEvaluator* method), 159  
**clear\_all\_tmp\_dirs()** (in module *easycv.utils.test\_util*), 296  
**clip\_coords()** (in module *easycv.utils.bbox\_util*), 288  
**close()** (*easycv.file.file\_io.OSSFile* method), 304  
**ClsDataset** (class in *easycv.datasets.classification*), 41  
**ClsDataset** (class in *easycv.datasets.classification.raw*), 64  
**ClsEvaluator** (class in *easycv.core.evaluation.classification\_eval*), 157  
**ClsHead** (class in *easycv.models.heads.cls\_head*), 251  
**ClsOdpsDataset** (class in *easycv.datasets.classification*), 41  
**ClsOdpsDataset** (class in *easycv.datasets.classification.odps*), 64  
**ClsSourceCifar10** (class in *easycv.datasets.classification.data\_sources*), 42  
**ClsSourceCifar10** (class in *easycv.datasets.classification.data\_sources.cifar*), 46  
**ClsSourceCifar100** (class in *easycv.datasets.classification.data\_sources*), 42  
**ClsSourceCifar100** (class in *easycv.datasets.classification.data\_sources.cifar*), 46  
**ClsSourceCifar100** (class in *easycv.datasets.classification.data\_sources.cifar*), 46  
**ClsSourceCUB** (class in *easycv.datasets.classification.data\_sources*), 44  
**ClsSourceImageList** (class in *easycv.datasets.classification.data\_sources*), 43  
**ClsSourceImageList** (class in *easycv.datasets.classification.data\_sources.image\_classification*), 47  
**ClsSourceImageListByClass** (class in *easycv.datasets.classification.data\_sources*), 42  
**ClsSourceImageListByClass** (class in *easycv.datasets.classification.data\_sources.classification*), 46  
**ClsSourceImageNetTFRecord** (class in *easycv.datasets.classification.data\_sources*), 42  
**ClsSourceImageNetTFRecord** (class in *easycv.datasets.classification.data\_sources.imagenet\_tfrecord*), 48  
**CocoDetectionEvaluator** (class in *easycv.core.evaluation.coco\_evaluation*), 158  
**COCOeval** (class in *easycv.core.evaluation.custom\_cocotools.cocoeval*), 155  
**COCOEvalWrapper** (class in *easycv.core.evaluation.coco\_tools*), 161  
**CocoMaskEvaluator** (class in *easycv.core.evaluation.coco\_evaluation*), 159  
**CoCoPoseTopDownEvaluator** (class in *easycv.core.evaluation.coco\_evaluation*), 160  
**COCOWrapper** (class in *easycv.core.evaluation.coco\_tools*), 161  
**Collect** (class in *easycv.datasets.shared.pipelines.format*), 116  
**collect\_env()** (in module *easycv.utils.collect\_env*), 289  
**collect\_results\_cpu()** (in module *easycv.apis.test*), 38  
**collect\_results\_gpu()** (in module *easycv.apis.test*), 38  
**ColorJitter** (class in *easycv.datasets.shared.pipelines.third\_transforms\_wrapper*), 118  
**colorspace** (*easycv.core.standard\_fields.TfExampleFields* attribute), 187, 188  
**ColorTransform** (class in *easycv.datasets.classification.pipelines.auto\_augment*), 62  
**compat\_dumps()** (in module *easycv.utils.json\_utils*), 293  
**Compose** (class in *easycv.datasets.shared.pipelines.transforms*), 126  
**compute\_ap()** (in module *easycv.core.evaluation.ap*), 156  
**compute\_average\_flops\_cost()** (in module *easycv.utils.flops\_counter*), 291  
**compute\_macs()** (*easycv.models.backbones.swin\_transformer\_dynamic* static method), 230  
**computeIoU()** (*easycv.core.evaluation.custom\_cocotools.cocoeval.COCOEvalWrapper* method), 155  
**ComputeMetrics()** (*easycv.core.evaluation.coco\_tools.COCOEvalWrapper* method), 162  
**computeOks()** (*easycv.core.evaluation.custom\_cocotools.cocoeval.COCOEvalWrapper* method), 155  
**concat\_all\_gather()** (in module *easycv.models.selfsup.moby*), 268

`concat_all_gather()` (in module `easycv.models.selfsup.moco`), 269  
`ConcatDataset` (class in `easycv.datasets.shared`), 112  
`ConcatDataset` (class in `easycv.datasets.shared.dataset_wrappers`), 129  
`ConditionalChannelWeighting` (class in `easycv.models.backbones.lightrnet`), 205  
`config_dict_edit()` (in module `easycv.utils.config_tools`), 290  
`Contrast` (class in `easycv.datasets.classification.pipelines.auto_augment`), 61  
`contrastive_loss()` (`easycv.models.selfsup.moby.MoBY` method), 267  
`ContrastiveHead` (class in `easycv.models.heads.contrastive_head`), 252  
`conv3x3()` (in module `easycv.models.backbones.xcit_transformer`), 239  
`conv_flops_counter_hook()` (in module `easycv.utils.flops_counter`), 292  
`conv_ws_2d()` (in module `easycv.models.utils.conv_ws`), 277  
`ConvDW` (class in `easycv.models.backbones.genet`), 194  
`ConvertImageDtype` (class in `easycv.datasets.shared.pipelines.third_transforms_converter`), 119  
`ConvKX` (class in `easycv.models.backbones.genet`), 194  
`ConvModule` (class in `easycv.models.utils.conv_module`), 276  
`ConvPatchEmbed` (class in `easycv.models.backbones.xcit_transformer`), 239  
`ConvWS2d` (class in `easycv.models.utils.conv_ws`), 277  
`copy()` (`easycv.file.base.IOBase` method), 299  
`copy()` (`easycv.file.base.IOLocal` method), 300  
`copy()` (`easycv.file.file_io.IO` method), 301  
`copy_attr()` (in module `easycv.utils.py_util`), 296  
`copytree()` (`easycv.file.base.IOBase` method), 299  
`copytree()` (`easycv.file.base.IOLocal` method), 300  
`copytree()` (`easycv.file.file_io.IO` method), 302  
`cosine_scheduler()` (in module `easycv.hooks.dino_hook`), 134  
`CosineSimilarity` (in module `easycv.utils.metric_distance`), 294  
`create_attn_mask()` (`easycv.models.backbones.swin_transformer_block` method), 231  
`create_from_str()` (`easycv.models.backbones.genet.AdaptiveAvgPool` static method), 193  
`create_from_str()` (`easycv.models.backbones.genet.BN` static method), 194  
`create_from_str()` (`easycv.models.backbones.genet.ConvDW` static method), 194  
`create_from_str()` (`easycv.models.backbones.genet.ConvKX` static method), 195  
`create_from_str()` (`easycv.models.backbones.genet.Flatten` static method), 195  
`create_from_str()` (`easycv.models.backbones.genet.Linear` static method), 195  
`create_from_str()` (`easycv.models.backbones.genet.MaxPool` static method), 196  
`create_from_str()` (`easycv.models.backbones.genet.MultiSumBlock` static method), 196  
`create_from_str()` (`easycv.models.backbones.genet.PlainNetBasicBlock` static method), 193  
`create_from_str()` (`easycv.models.backbones.genet.RELU` static method), 197  
`create_from_str()` (`easycv.models.backbones.genet.ResBlock` static method), 197  
`create_from_str()` (`easycv.models.backbones.genet.Sequential` static method), 197  
`create_from_str()` (`easycv.models.backbones.genet.SuperResK1DW` static method), 199  
`create_from_str()` (`easycv.models.backbones.genet.SuperResK1DWK1` static method), 199  
`create_from_str()` (`easycv.models.backbones.genet.SuperResK1KX` static method), 198  
`create_from_str()` (`easycv.models.backbones.genet.SuperResK1KXK1` static method), 199  
`create_from_str()` (`easycv.models.backbones.genet.SuperResKXXK` static method), 198  
`create_from_str()` (`easycv.models.backbones.genet.SuperResKXXK1` static method), 198  
`create_namedtuple()` (in module `easycv.file.utils`), 304  
`CrossEntropyLossWithLabelSmooth` (class in `easycv.models.loss.pytorch_metric_learning`), 256  
`CrossResolutionWeighting` (class in `easycv.models.backbones.lightrnet`), 205  
`CSPDarknet` (class in `easycv.models.backbones.darknet`), 192  
`CSPLayer` (class in `easycv.models.backbones.network_blocks`), 214  
`cuda()` (`easycv.utils.alias_multinomial.AliasMethod` method), 287  
`cumsum_length()` (`easycv.datasets.shared.data_sources.concat.SourceConcat` method), 114  
`cumsum_length()` (`easycv.datasets.shared.data_sources.SourceConcat` method), 114  
`cumulative_sizes` (`easycv.datasets.shared.ConcatDataset` attribute), 112  
`current_image_sizes` (`easycv.datasets.shared.dataset_wrappers.ConcatDataset` attribute), 129  
`current_image_sizes` (`easycv.runner.ev_runner.EVRunner` method), 305  
`Cutout` (class in `easycv.datasets.classification.pipelines.auto_augment`), 63  
`DaViColorTwist` (class in `easycv.datasets.classification.pipelines.auto_augment`), 63

`easycv.datasets.shared.pipelines.dali_transforms)`, `deconv_flops_counter_hook()` (in module `easycv.utils.flops_counter`), 292  
**DaliCropMirrorNormalize** (class in `default()` (`easycv.utils.json_utils.MyEncoder` method), `easycv.datasets.shared.pipelines.dali_transforms`), 292  
**DaliImageDecoder** (class in `easycv.datasets.shared.pipelines.format`), `easycv.datasets.shared.pipelines.dali_transforms`), 117  
**DaliImageNetTFRecordDataSet** (class in `method`), 127  
**DaliImageNetTFRecordDataSet** (class in `define_graph()` (`easycv.datasets.shared.dali_tfrecord_imagenet.ImageNetTFRecordDataSet` method), 127  
**DaliImageNetTFRecordDataSet** (class in `define_graph()` (`easycv.datasets.shared.dali_tfrecord_multi_view.DaliImageNetTFRecordDataSet` method), 128  
**DaliLoaderWrapper** (class in `depth2blocks` (`easycv.models.backbones.darknet.Darknet` attribute), 192  
**DaliLoaderWrapper** (class in `DetDataset` (class in `easycv.datasets.detection`), 65  
**DaliLoaderWrapper** (class in `DetDataset` (class in `easycv.datasets.detection.raw`), 95  
**DaliLoaderWrapper** (class in `easycv.datasets.shared.dali_tfrecord_multi_view`), `detection_bbox_xmax` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**DaliRandomGrayscale** (class in `easycv.datasets.shared.pipelines.dali_transforms`), `detection_bbox_xmin` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**DaliRandomResizedCrop** (class in `easycv.datasets.shared.pipelines.dali_transforms`), `detection_bbox_ymax` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**DaliResize** (class in `easycv.datasets.shared.pipelines.dali_transforms`), `detection_bbox_ymin` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**DaliTFRecordMultiViewDataSet** (class in `easycv.datasets.shared.dali_tfrecord_multi_view`), `detection_boundaries` (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
**DaliTFRecordMultiViewPipe** (class in `easycv.datasets.shared.dali_tfrecord_multi_view`), `detection_boxes` (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
**Darknet** (class in `easycv.models.backbones.darknet`), `detection_class_label` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**dataloader** (`easycv.hooks.eval_hook.DistEvalHook` attribute), 135  
**dataloader** (`easycv.hooks.eval_hook.EvalHook` attribute), 135  
**dataset** (`easycv.datasets.loader.build_loader.InfiniteDataLoader` attribute), 98  
**dataset\_name** (`easycv.core.standard_fields.InputDataFields` attribute), 185  
**DatasetInfo** (class in `easycv.datasets.pose.data_sources.top_down`), `detection_masks` (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
**datasets** (`easycv.datasets.shared.ConcatDataset` attribute), 112  
**datasets** (`easycv.datasets.shared.dataset_wrappers.ConcatDataset` attribute), 129  
**DebiasedContrastiveHead** (class in `easycv.models.heads.contrastive_head`), 252  
**decode()** (`easycv.models.pose.heads.topdown_heatmap_base.TopdownHeatmapBaseHead` class method), 259  
**decode\_outputs()** (`easycv.models.detection.yolox.yolo_head.YOLOXHead` class method), 249  
**DefaultFormatBundle** (class in `method`), 127  
**define\_graph()** (`easycv.datasets.shared.dali_tfrecord_imagenet.ImageNetTFRecordDataSet` method), 127  
**define\_graph()** (`easycv.datasets.shared.dali_tfrecord_multi_view.DaliImageNetTFRecordDataSet` method), 128  
**depth2blocks** (`easycv.models.backbones.darknet.Darknet` attribute), 192  
**DetDataset** (class in `easycv.datasets.detection`), 65  
**DetDataset** (class in `easycv.datasets.detection.raw`), 95  
**detection\_bbox\_xmax** (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**detection\_bbox\_xmin** (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**detection\_bbox\_ymax** (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**detection\_bbox\_ymin** (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**detection\_boundaries** (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
**detection\_boxes** (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
**detection\_class\_label** (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**detection\_classes** (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
**detection\_keypoints** (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
**detection\_masks** (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
**detection\_score** (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
**detection\_scores** (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
**DetectionResultFields** (class in `easycv.core.standard_fields`), 186  
**DetImagesMixDataSet** (class in `easycv.datasets.detection`), 66  
**DetImagesMixDataSet** (class in `easycv.datasets.detection.mix`), 93  
**DetImageToMixDataset** (class in `easycv.datasets.detection.data_sources`), 93  
**DetImageToMixDataset** (class in `easycv.datasets.detection.data_sources`), 93

|                                                     |                                                     |        |                                                                     |                                                    |        |
|-----------------------------------------------------|-----------------------------------------------------|--------|---------------------------------------------------------------------|----------------------------------------------------|--------|
| 67                                                  |                                                     |        |                                                                     | easycv.datasets.loader.sampler), 98                |        |
| DetSourceCoco                                       | (class                                              | in     | DistributeMSELoss                                                   | (class                                             | in     |
|                                                     | easycv.datasets.detection.data_sources.coco),       |        |                                                                     | easycv.models.loss.pytorch_metric_learning),       |        |
| 71                                                  |                                                     |        | 256                                                                 |                                                    |        |
| DetSourcePAI                                        | (class                                              | in     | DotproductSimilarity()                                              | (in                                                | module |
|                                                     | easycv.datasets.detection.data_sources),            |        |                                                                     | easycv.utils.metric_distance), 294                 |        |
| 68                                                  |                                                     |        | download_tfrecord()                                                 | (in                                                | module |
| DetSourcePAI                                        | (class                                              | in     | easycv.datasets.utils.tfrecord_util), 130                           |                                                    |        |
|                                                     | easycv.datasets.detection.data_sources.pai_format), |        | draw()                                                              | (easycv.utils.alias_multinomial.AliasMethod        |        |
| 72                                                  |                                                     |        | method), 287                                                        |                                                    |        |
| DetSourceRaw                                        | (class                                              | in     | drop_last(easycv.datasets.loader.build_loader.InfiniteDataLoader    |                                                    |        |
|                                                     | easycv.datasets.detection.data_sources),            |        | attribute), 98                                                      |                                                    |        |
| 69                                                  |                                                     |        | drop_path()                                                         | (in                                                | module |
| DetSourceRaw                                        | (class                                              | in     | easycv.models.backbones.vit_transformer_dynamic),                   |                                                    |        |
|                                                     | easycv.datasets.detection.data_sources.raw),        |        | 236                                                                 |                                                    |        |
| 73                                                  |                                                     |        | DropBlock2D                                                         | (class                                             | in     |
| DetSourceVOC                                        | (class                                              | in     | easycv.models.backbones.resnet), 217                                |                                                    |        |
|                                                     | easycv.datasets.detection.data_sources),            |        | DropPath(class in easycv.models.backbones.vit_transformer_dynamic), |                                                    |        |
| 70                                                  |                                                     |        | 236                                                                 |                                                    |        |
| DetSourceVOC                                        | (class                                              | in     | dump() (in module easycv.utils.json_utils), 293                     |                                                    |        |
|                                                     | easycv.datasets.detection.data_sources.voc), 74     |        | dumps() (in module easycv.utils.json_utils), 293                    |                                                    |        |
| dilation                                            | (easycv.models.utils.conv_ws.ConvWS2d at-           |        | DWConv(class in easycv.models.backbones.network_blocks),            |                                                    |        |
| tribute), 277                                       |                                                     |        | 213                                                                 |                                                    |        |
| DINO(class in easycv.models.selfsup.dino), 264      |                                                     |        | dynamic_deit_small_p16()                                            | (in                                                | module |
| DINOHook(class in easycv.hooks.dino_hook), 134      |                                                     |        |                                                                     | easycv.models.backbones.vit_transformer_dynamic),  |        |
| DINOLoss(class in easycv.models.selfsup.dino), 264  |                                                     |        | 238                                                                 |                                                    |        |
| DINONeck(class in easycv.models.selfsup.necks), 269 |                                                     |        | dynamic_deit_tiny_p16()                                             | (in                                                | module |
| dist_exec_wrapper()                                 | (in                                                 | module | easycv.models.backbones.vit_transformer_dynamic),                   |                                                    |        |
|                                                     | easycv.utils.test_util), 296                        |        | 238                                                                 |                                                    |        |
| dist_forward_collect()                              | (in                                                 | module | dynamic_k_matching()                                                |                                                    |        |
|                                                     | easycv.utils.collect), 289                          |        | (easycv.models.detection.yolox.yolo_head.YOLOXHead                  |                                                    |        |
| dist_zero_exec()                                    | (in module easycv.utils.dist_utils),                |        | method), 249                                                        |                                                    |        |
| 290                                                 |                                                     |        | dynamic_swin_base_p4_w7_224()                                       | (in                                                | module |
| DistEvalHook(class in easycv.hooks.eval_hook), 135  |                                                     |        |                                                                     | easycv.models.backbones.swin_transformer_dynamic), |        |
| distill_loss()                                      | (in                                                 | module | 235                                                                 |                                                    |        |
|                                                     | easycv.models.classification.classification),       |        | dynamic_swin_small_p4_w7_224()                                      | (in                                                | module |
| 244                                                 |                                                     |        |                                                                     | easycv.models.backbones.swin_transformer_dynamic), |        |
| distributed_sinkhorn()                              | (in                                                 | module | 235                                                                 |                                                    |        |
|                                                     | easycv.models.selfsup.swav), 275                    |        | dynamic_swin_tiny_p4_w7_224()                                       | (in                                                | module |
| DistributedGivenIterationSampler                    | (class                                              | in     | easycv.models.backbones.swin_transformer_dynamic),                  |                                                    |        |
|                                                     | easycv.datasets.loader), 97                         |        | 235                                                                 |                                                    |        |
| DistributedGivenIterationSampler                    | (class                                              | in     | dynamic_vit_base_p16()                                              | (in                                                | module |
|                                                     | easycv.datasets.loader.sampler), 99                 |        |                                                                     | easycv.models.backbones.vit_transformer_dynamic),  |        |
| DistributedGroupSampler                             | (class                                              | in     | 239                                                                 |                                                    |        |
|                                                     | easycv.datasets.loader), 96                         |        | dynamic_vit_huge_p14()                                              | (in                                                | module |
| DistributedGroupSampler                             | (class                                              | in     | easycv.models.backbones.vit_transformer_dynamic),                   |                                                    |        |
|                                                     | easycv.datasets.loader.sampler), 99                 |        | 239                                                                 |                                                    |        |
| DistributedLossWrapper                              | (class                                              | in     | dynamic_vit_large_p16()                                             | (in                                                | module |
|                                                     | easycv.models.utils.dist_utils), 278                |        |                                                                     | easycv.models.backbones.vit_transformer_dynamic),  |        |
| DistributedMinerWrapper                             | (class                                              | in     | 239                                                                 |                                                    |        |
|                                                     | easycv.models.utils.dist_utils), 278                |        |                                                                     |                                                    |        |
| DistributedMPSSampler                               | (class                                              | in     | E                                                                   |                                                    |        |
|                                                     | easycv.datasets.loader.sampler), 98                 |        | easycv                                                              |                                                    |        |
| DistributedSampler                                  | (class                                              | in     | module, 299                                                         |                                                    |        |



---

```

easy cv.apis
 module, 37
easy cv.apis.export
 module, 37
easy cv.apis.test
 module, 37
easy cv.apis.train
 module, 38
easy cv.apis.train_misc
 module, 39
easy cv.core
 module, 155
easy cv.core.evaluation
 module, 155
easy cv.core.evaluation.ap
 module, 156
easy cv.core.evaluation.auc_eval
 module, 157
easy cv.core.evaluation.base_evaluator
 module, 157
easy cv.core.evaluation.builder
 module, 157
easy cv.core.evaluation.classification_eval
 module, 157
easy cv.core.evaluation.coco_evaluation
 module, 158
easy cv.core.evaluation.coco_tools
 module, 160
easy cv.core.evaluation.custom_cocotools
 module, 155
easy cv.core.evaluation.custom_cocotools.coco_eval
 module, 155
easy cv.core.evaluation.faceid_pair_eval
 module, 167
easy cv.core.evaluation.metric_registry
 module, 168
easy cv.core.evaluation.mse_eval
 module, 168
easy cv.core.evaluation.retrival_topk_eval
 module, 169
easy cv.core.evaluation.top_down_eval
 module, 169
easy cv.core.optimizer
 module, 172
easy cv.core.optimizer.lars
 module, 172
easy cv.core.optimizer.ranger
 module, 173
easy cv.core.post_processing
 module, 173
easy cv.core.post_processing.nms
 module, 177
easy cv.core.post_processing.pose_transforms
 module, 178
easy cv.core.standard_fields
 module, 184
easy cv.core.visualization
 module, 181
easy cv.core.visualization.image
 module, 182
easy cv.datasets
 module, 41
easy cv.datasets.builder
 module, 131
easy cv.datasets.classification
 module, 41
easy cv.datasets.classification.data_sources
 module, 42
easy cv.datasets.classification.data_sources.cifar
 module, 46
easy cv.datasets.classification.data_sources.class_list
 module, 46
easy cv.datasets.classification.data_sources.fashiongen_h5
 module, 47
easy cv.datasets.classification.data_sources.image_list
 module, 47
easy cv.datasets.classification.data_sources.imagenet_tfrec
 module, 48
easy cv.datasets.classification.data_sources.utils
 module, 48
easy cv.datasets.classification.odps
 module, 64
easy cv.datasets.classification.pipelines
 module, 50
easy cv.datasets.classification.pipelines.auto_augment
 module, 54
easy cv.datasets.classification.pipelines.transform
 module, 63
easy cv.datasets.classification.raw
 module, 64
easy cv.datasets.detection
 module, 65
easy cv.datasets.detection.data_sources
 module, 67
easy cv.datasets.detection.data_sources.coco
 module, 71
easy cv.datasets.detection.data_sources.pai_format
 module, 72
easy cv.datasets.detection.data_sources.raw
 module, 73
easy cv.datasets.detection.data_sources.utils
 module, 74
easy cv.datasets.detection.data_sources.voc
 module, 74
easy cv.datasets.detection.mix
 module, 93
easy cv.datasets.detection.pipelines
 module, 75

```

|                                                          |                                                            |
|----------------------------------------------------------|------------------------------------------------------------|
| easy cv.datasets.detection.pipelines.mn_transforms       | easy cv.datasets.shared.multi_view                         |
| module, 84                                               | module, 129                                                |
| easy cv.datasets.detection.raw                           | easy cv.datasets.shared.odps_reader                        |
| module, 95                                               | module, 129                                                |
| easy cv.datasets.loader                                  | easy cv.datasets.shared.pipelines                          |
| module, 96                                               | module, 115                                                |
| easy cv.datasets.loader.build_loader                     | easy cv.datasets.shared.pipelines.dali_transforms          |
| module, 97                                               | module, 115                                                |
| easy cv.datasets.loader.sampler                          | easy cv.datasets.shared.pipelines.format                   |
| module, 98                                               | module, 116                                                |
| easy cv.datasets.pose                                    | easy cv.datasets.shared.pipelines.third_transforms_wrapper |
| module, 100                                              | module, 118                                                |
| easy cv.datasets.pose.data_sources                       | easy cv.datasets.shared.pipelines.transforms               |
| module, 100                                              | module, 126                                                |
| easy cv.datasets.pose.data_sources.coco                  | easy cv.datasets.shared.raw                                |
| module, 102                                              | module, 130                                                |
| easy cv.datasets.pose.data_sources.top_down              | easy cv.datasets.utils                                     |
| module, 103                                              | module, 130                                                |
| easy cv.datasets.pose.pipelines                          | easy cv.datasets.utils.tfrecord_util                       |
| module, 103                                              | module, 130                                                |
| easy cv.datasets.pose.pipelines.transforms               | easy cv.datasets.utils.type_util                           |
| module, 106                                              | module, 131                                                |
| easy cv.datasets.pose.top_down                           | easy cv.file                                               |
| module, 109                                              | module, 299                                                |
| easy cv.datasets.registry                                | easy cv.file.base                                          |
| module, 131                                              | module, 299                                                |
| easy cv.datasets.selfsup                                 | easy cv.file.file_io                                       |
| module, 109                                              | module, 300                                                |
| easy cv.datasets.selfsup.data_sources                    | easy cv.file.utils                                         |
| module, 109                                              | module, 304                                                |
| easy cv.datasets.selfsup.data_sources.image_list         | easy cv.hooks                                              |
| module, 110                                              | module, 133                                                |
| easy cv.datasets.selfsup.data_sources.imagenet_extractor | easy cv.hooks.best_ckpt_saver_hook                         |
| module, 110                                              | module, 133                                                |
| easy cv.datasets.selfsup.pipelines                       | easy cv.hooks.builder                                      |
| module, 111                                              | module, 133                                                |
| easy cv.datasets.selfsup.pipelines.transforms            | easy cv.hooks.byol_hook                                    |
| module, 111                                              | module, 134                                                |
| easy cv.datasets.shared                                  | easy cv.hooks.dino_hook                                    |
| module, 112                                              | module, 134                                                |
| easy cv.datasets.shared.base                             | easy cv.hooks.ema_hook                                     |
| module, 126                                              | module, 134                                                |
| easy cv.datasets.shared.dali_tfrecord_imagenet           | easy cv.hooks.eval_hook                                    |
| module, 127                                              | module, 135                                                |
| easy cv.datasets.shared.dali_tfrecord_multi_view         | easy cv.hooks.export_hook                                  |
| module, 128                                              | module, 136                                                |
| easy cv.datasets.shared.data_sources                     | easy cv.hooks.extractor                                    |
| module, 114                                              | module, 137                                                |
| easy cv.datasets.shared.data_sources.concat              | easy cv.hooks.optimizer_hook                               |
| module, 114                                              | module, 137                                                |
| easy cv.datasets.shared.data_sources.image_npy           | easy cv.hooks.oss_sync_hook                                |
| module, 114                                              | module, 137                                                |
| easy cv.datasets.shared.dataset_wrappers                 | easy cv.hooks.registry                                     |
| module, 129                                              | module, 138                                                |

---

```

easy cv . hooks . show_time_hook
 module, 138
easy cv . hooks . swav_hook
 module, 138
easy cv . hooks . sync_norm_hook
 module, 139
easy cv . hooks . sync_random_size_hook
 module, 139
easy cv . hooks . tensorboard
 module, 140
easy cv . hooks . wandb
 module, 140
easy cv . hooks . yolox_lr_hook
 module, 140
easy cv . hooks . yolox_mode_switch_hook
 module, 141
easy cv . models
 module, 191
easy cv . models . backbones
 module, 191
easy cv . models . backbones . benchmark_mlp
 module, 191
easy cv . models . backbones . bninception
 module, 191
easy cv . models . backbones . darknet
 module, 192
easy cv . models . backbones . genet
 module, 193
easy cv . models . backbones . hrnet
 module, 200
easy cv . models . backbones . inceptionv3
 module, 204
easy cv . models . backbones . lightrnet
 module, 204
easy cv . models . backbones . mae_vit_transformer
 module, 210
easy cv . models . backbones . mnasnet
 module, 211
easy cv . models . backbones . mobilenetv2
 module, 211
easy cv . models . backbones . network_blocks
 module, 212
easy cv . models . backbones . pytorch_image_models_wrapper
 module, 215
easy cv . models . backbones . resnest
 module, 216
easy cv . models . backbones . resnet
 module, 219
easy cv . models . backbones . resnet_jit
 module, 221
easy cv . models . backbones . resnext
 module, 224
easy cv . models . backbones . shuffle_transformer
 module, 225
easy cv . models . backbones . swin_transformer_dynamic
 module, 229
easy cv . models . backbones . vit_transformer_dynamic
 module, 236
easy cv . models . backbones . xcit_transformer
 module, 239
easy cv . models . base
 module, 284
easy cv . models . builder
 module, 285
easy cv . models . classification
 module, 244
easy cv . models . classification . classification
 module, 244
easy cv . models . classification . necks
 module, 245
easy cv . models . detection
 module, 248
easy cv . models . detection . utils
 module, 248
easy cv . models . detection . utils . boxes
 module, 248
easy cv . models . detection . yolox
 module, 248
easy cv . models . detection . yolox . yolo_head
 module, 248
easy cv . models . detection . yolox . yolo_pafpn
 module, 249
easy cv . models . detection . yolox . yolox
 module, 250
easy cv . models . detection . yolox_edge
 module, 251
easy cv . models . detection . yolox_edge . yolox_edge
 module, 251
easy cv . models . heads
 module, 251
easy cv . models . heads . cls_head
 module, 251
easy cv . models . heads . contrastive_head
 module, 252
easy cv . models . heads . latent_pred_head
 module, 253
easy cv . models . heads . mp_metric_head
 module, 253
easy cv . models . heads . multi_cls_head
 module, 254
easy cv . models . loss
 module, 255
easy cv . models . loss . iou_loss
 module, 255
easy cv . models . loss . mse_loss
 module, 255
easy cv . models . loss . pytorch_metric_learning
 module, 256

```

|                                                                   |                                                     |
|-------------------------------------------------------------------|-----------------------------------------------------|
| <code>easycv.models.modelzoo</code>                               | <code>easycv.models.utils.ops</code>                |
| module, 285                                                       | module, 282                                         |
| <code>easycv.models.pose</code>                                   | <code>easycv.models.utils.pos_embed</code>          |
| module, 258                                                       | module, 282                                         |
| <code>easycv.models.pose.heads</code>                             | <code>easycv.models.utils.res_layer</code>          |
| module, 258                                                       | module, 282                                         |
| <code>easycv.models.pose.heads.topdown_heatmap_base_head</code>   | <code>easycv.models.utils.scale</code>              |
| module, 259                                                       | module, 283                                         |
| <code>easycv.models.pose.heads.topdown_heatmap_simple_head</code> | <code>easycv.models.utils.sobel</code>              |
| module, 259                                                       | module, 283                                         |
| <code>easycv.models.pose.top_down</code>                          | <code>easycv.predictors</code>                      |
| module, 261                                                       | module, 143                                         |
| <code>easycv.models.registry</code>                               | <code>easycv.predictors.base</code>                 |
| module, 285                                                       | module, 143                                         |
| <code>easycv.models.selfsup</code>                                | <code>easycv.predictors.builder</code>              |
| module, 263                                                       | module, 143                                         |
| <code>easycv.models.selfsup.byol</code>                           | <code>easycv.predictors.classifier</code>           |
| module, 263                                                       | module, 144                                         |
| <code>easycv.models.selfsup.dino</code>                           | <code>easycv.predictors.detector</code>             |
| module, 263                                                       | module, 145                                         |
| <code>easycv.models.selfsup.mae</code>                            | <code>easycv.predictors.feature_extractor</code>    |
| module, 265                                                       | module, 147                                         |
| <code>easycv.models.selfsup.mixco</code>                          | <code>easycv.predictors.interface</code>            |
| module, 266                                                       | module, 150                                         |
| <code>easycv.models.selfsup.moby</code>                           | <code>easycv.predictors.pose_predictor</code>       |
| module, 266                                                       | module, 152                                         |
| <code>easycv.models.selfsup.moco</code>                           | <code>easycv.runner</code>                          |
| module, 268                                                       | module, 305                                         |
| <code>easycv.models.selfsup.necks</code>                          | <code>easycv.runner.ev_runner</code>                |
| module, 269                                                       | module, 305                                         |
| <code>easycv.models.selfsup.simclr</code>                         | <code>easycv.toolkit</code>                         |
| module, 273                                                       | module, 306                                         |
| <code>easycv.models.selfsup.swav</code>                           | <code>easycv.toolkit.prune</code>                   |
| module, 274                                                       | module, 306                                         |
| <code>easycv.models.utils</code>                                  | <code>easycv.toolkit.quantize</code>                |
| module, 275                                                       | module, 306                                         |
| <code>easycv.models.utils.accuracy</code>                         | <code>easycv.toolkit.quantize.quantize_utils</code> |
| module, 275                                                       | module, 306                                         |
| <code>easycv.models.utils.activation</code>                       | <code>easycv.utils</code>                           |
| module, 275                                                       | module, 287                                         |
| <code>easycv.models.utils.conv_module</code>                      | <code>easycv.utils.alias_multinomial</code>         |
| module, 276                                                       | module, 287                                         |
| <code>easycv.models.utils.conv_ws</code>                          | <code>easycv.utils.bbox_util</code>                 |
| module, 277                                                       | module, 287                                         |
| <code>easycv.models.utils.dist_utils</code>                       | <code>easycv.utils.checkpoint</code>                |
| module, 278                                                       | module, 288                                         |
| <code>easycv.models.utils.gather_layer</code>                     | <code>easycv.utils.collect</code>                   |
| module, 278                                                       | module, 289                                         |
| <code>easycv.models.utils.init_weights</code>                     | <code>easycv.utils.collect_env</code>               |
| module, 279                                                       | module, 289                                         |
| <code>easycv.models.utils.multi_pooling</code>                    | <code>easycv.utils.config_tools</code>              |
| module, 279                                                       | module, 289                                         |
| <code>easycv.models.utils.norm</code>                             | <code>easycv.utils.constant</code>                  |
| module, 280                                                       | module, 290                                         |

easycv.utils.dist\_utils  
     module, 290  
 easycv.utils.eval\_utils  
     module, 291  
 easycv.utils.flops\_counter  
     module, 291  
 easycv.utils.gather  
     module, 292  
 easycv.utils.json\_utils  
     module, 292  
 easycv.utils.logger  
     module, 294  
 easycv.utils.metric\_distance  
     module, 294  
 easycv.utils.misc  
     module, 295  
 easycv.utils.preprocess\_function  
     module, 295  
 easycv.utils.profiling  
     module, 295  
 easycv.utils.py\_util  
     module, 296  
 easycv.utils.registry  
     module, 296  
 easycv.utils.test\_util  
     module, 296  
 easycv.utils.user\_config\_params\_utils  
     module, 297  
 easycv.version  
     module, 307  
 EMAHook (class in easycv.hooks.ema\_hook), 134  
 EmbeddingExpansion() (in module  
     easycv.models.heads.mp\_metric\_head), 253  
 empty\_flops\_counter\_hook() (in module  
     easycv.utils.flops\_counter), 292  
 Equalize (class in easycv.datasets.classification.pipelines.augmentation), 60  
 EvalHook (class in easycv.hooks.eval\_hook), 135  
 evaluate() (easycv.core.evaluation.base\_evaluator.Evaluator  
     method), 157  
 evaluate() (easycv.core.evaluation.custom\_cocotools.cocoeval.COCOEvaluator  
     method), 155  
 evaluate() (easycv.datasets.classification.ClsDataset  
     method), 41  
 evaluate() (easycv.datasets.classification.ClsOdpsDataset  
     method), 42  
 evaluate() (easycv.datasets.classification.odps.ClsOdpsDataset  
     method), 64  
 evaluate() (easycv.datasets.classification.raw.ClsDataset  
     method), 64  
 evaluate() (easycv.datasets.detection.DetDataset  
     method), 65  
 evaluate() (easycv.datasets.detection.raw.DetDataset  
     method), 95  
 evaluate() (easycv.datasets.pose.PoseTopDownDataset  
     method), 100  
 evaluate() (easycv.datasets.pose.top\_down.PoseTopDownDataset  
     method), 109  
 evaluate() (easycv.datasets.shared.base.BaseDataset  
     method), 126  
 evaluate() (easycv.datasets.shared.BaseDataset  
     method), 113  
 evaluate() (easycv.datasets.shared.dali\_tfrecord\_imagenet.DaliLoaderWrapper  
     method), 127  
 evaluate() (easycv.datasets.shared.multi\_view.MultiViewDataset  
     method), 129  
 evaluate() (easycv.datasets.shared.MultiViewDataset  
     method), 113  
 evaluate() (easycv.datasets.shared.raw.RawDataset  
     method), 130  
 evaluate() (easycv.datasets.shared.RawDataset  
     method), 113  
 evaluate() (easycv.hooks.eval\_hook.EvalHook  
     method), 135  
 evaluateImg() (easycv.core.evaluation.custom\_cocotools.cocoeval.COCOEvaluator  
     method), 155  
 Evaluator (class in easycv.core.evaluation.base\_evaluator),  
     157  
 EVRunner (class in easycv.runner.ev\_runner), 305  
 exif\_size() (in module  
     easycv.datasets.detection.data\_sources.utils),  
     74  
 exists() (easycv.file.base.IOBase method), 299  
 exists() (easycv.file.base.IOLocal method), 300  
 exists() (easycv.file.file\_io.IO method), 301  
 expansion (easycv.models.backbones.resnest.Bottleneck  
     attribute), 217  
 expansion (easycv.models.backbones.resnet.BasicBlock  
     attribute), 219  
 expansion (easycv.models.backbones.resnet.Bottleneck  
     attribute), 219  
 expansion (easycv.models.backbones.resnet\_jit.BasicBlock  
     attribute), 221  
 expansion (easycv.models.backbones.resnet\_jit.Bottleneck  
     attribute), 222  
 export() (in module easycv.apis.export), 37  
 export\_model() (easycv.hooks.export\_hook.ExportHook  
     method), 136  
 ExportDetectionsToCOCO() (in module  
     easycv.core.evaluation.coco\_tools), 165  
 ExportGroundtruthToCOCO() (in module  
     easycv.core.evaluation.coco\_tools), 164  
 ExportHook (class in easycv.hooks.export\_hook), 136  
 ExportKeypointsToCOCO() (in module  
     easycv.core.evaluation.coco\_tools), 167  
 ExportSegmentsToCOCO() (in module  
     easycv.core.evaluation.coco\_tools), 166  
 ExportSingleImageDetectionBoxesToCoco() (in

[module](#) [easycv.core.evaluation.coco\\_tools](#)), [flip\\_back\(\)](#) (in [module](#) [easycv.core.post\\_processing.pose\\_transforms](#)),  
[164](#) [179](#)  
[ExportSingleImageDetectionMasksToCoco\(\)](#) (in [module](#) [easycv.core.evaluation.coco\\_tools](#)), [fliplr\\_joints\(\)](#) (in [module](#) [easycv.core.post\\_processing](#)), [173](#)  
[165](#)  
[ExportSingleImageGroundtruthToCoco\(\)](#) (in [module](#) [easycv.core.evaluation.coco\\_tools](#)), [163](#) [fliplr\\_joints\(\)](#) (in [module](#) [easycv.core.post\\_processing.pose\\_transforms](#)),  
[extra\\_repr\(\)](#) ([easycv.models.backbones.shuffle\\_transformer.PatchMerging](#) [method](#)), [227](#) [fliplr\\_regression\(\)](#) (in [module](#) [easycv.core.post\\_processing](#)), [174](#)  
[extra\\_repr\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.BasicLayer](#) [method](#)), [233](#) [fliplr\\_regression\(\)](#) (in [module](#) [easycv.core.post\\_processing.pose\\_transforms](#)),  
[extra\\_repr\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.PatchMerging](#) [method](#)), [232](#) [178](#)  
[extra\\_repr\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.BasicLayer](#) [method](#)), [232](#) [flops\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.BasicLayer](#) [method](#)), [233](#)  
[extra\\_repr\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.PatchEmbed](#) [method](#)), [230](#) [flops\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.PatchEmbed](#) [method](#)), [234](#)  
[Extractor](#) (class in [easycv.hooks.extractor](#)), [137](#) [flops\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.PatchMerging](#) [method](#)), [232](#)  
**F** [flops\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.SwinTransformer](#) [method](#)), [235](#)  
[faceid\\_evaluate\(\)](#) (in [module](#) [easycv.core.evaluation.faceid\\_pair\\_eval](#)), [167](#) [flops\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.SwinTransformer](#) [method](#)), [232](#)  
[FaceIDNeck](#) (class in [easycv.models.classification.necks](#)), [246](#) [flops\(\)](#) ([easycv.models.backbones.swin\\_transformer\\_dynamic.WindowAttention](#) [method](#)), [230](#)  
[FaceIDPairEvaluator](#) (class in [easycv.core.evaluation.faceid\\_pair\\_eval](#)), [168](#) [flops\\_to\\_string\(\)](#) (in [module](#) [easycv.utils.flops\\_counter](#)), [291](#)  
[FashionGenH5](#) (class in [easycv.datasets.classification.data\\_sources.fashiongen\\_h5](#)), [47](#) [flush\(\)](#) ([easycv.file.file\\_io.OSSFile](#) [method](#)), [304](#)  
[FEAT\\_CHANNELS](#) ([easycv.models.heads.multi\\_cls\\_head.MultiClsHead](#) [attribute](#)), [254](#) [flush\\_buffer](#) ([easycv.hooks.eval\\_hook.EvalHook](#) [attribute](#)), [135](#)  
[FEAT\\_LAST\\_UNPOOL](#) ([easycv.models.heads.multi\\_cls\\_head.MultiClsHead](#) [attribute](#)), [254](#) [FocalLoss2d](#) (class in [easycv.models.loss.pytorch\\_metric\\_learning](#)), [256](#)  
[features\(\)](#) ([easycv.models.backbones.bninception.BNInception](#) [method](#)), [191](#) [flops\(\)](#) ([easycv.models.backbones.network\\_blocks](#)), [215](#)  
[filename](#) ([easycv.core.standard\\_fields.InputDataFields](#) [attribute](#)), [184](#), [185](#) [format\\_results\(\)](#) ([easycv.datasets.detection.DetImagesMixDataset](#) [method](#)), [67](#)  
[filename](#) ([easycv.core.standard\\_fields.TfExampleFields](#) [attribute](#)), [187](#), [188](#) [format\\_results\(\)](#) ([easycv.datasets.detection.mix.DetImagesMixDataset](#) [method](#)), [94](#)  
[filter\\_annotations\(\)](#) ([easycv.core.evaluation.custom\\_cocotools.cocoeval.COCOEvaluator](#) [method](#)), [156](#) [forward\(\)](#) ([easycv.datasets.shared.pipelines.third\\_transforms\\_wrapper.AutoAugmentWrapper](#) [method](#)), [118](#)  
[filter\\_gt\\_bboxes\(\)](#) ([easycv.datasets.detection.pipelines.mm\\_transforms.MMRandomAffine](#) [method](#)), [87](#) [forward\(\)](#) ([easycv.datasets.shared.pipelines.third\\_transforms\\_wrapper.ComposeWrapper](#) [method](#)), [118](#)  
[filter\\_gt\\_bboxes\(\)](#) ([easycv.datasets.detection.pipelines.MMRandomAffine](#) [method](#)), [78](#) [forward\(\)](#) ([easycv.datasets.shared.pipelines.third\\_transforms\\_wrapper.ComposeWrapper](#) [method](#)), [119](#)  
[FiveCrop](#) (class in [easycv.datasets.shared.pipelines.third\\_transforms\\_wrapper](#)), [119](#) [forward\(\)](#) ([easycv.datasets.shared.pipelines.third\\_transforms\\_wrapper.ComposeWrapper](#) [method](#)), [119](#)  
[Flatten](#) (class in [easycv.models.backbones.genet](#)), [195](#) [forward\(\)](#) ([easycv.datasets.shared.pipelines.third\\_transforms\\_wrapper.ComposeWrapper](#) [method](#)), [119](#)  
[flip\\_back\(\)](#) (in [module](#) [easycv.core.post\\_processing](#)), [173](#) [forward\(\)](#) ([easycv.datasets.shared.pipelines.third\\_transforms\\_wrapper.ComposeWrapper](#) [method](#)), [119](#)



Index 337

method), 205

forward() (easycv.models.backbones.lightrnet.Stem method), 207

forward() (easycv.models.backbones.mae\_vit\_transformer.DynamicViT method), 210

forward() (easycv.models.backbones.mnasnet.MNASNet method), 211

forward() (easycv.models.backbones.mobilenetv2.MobileNetV2 method), 211

forward() (easycv.models.backbones.network\_blocks.BaseConv method), 212

forward() (easycv.models.backbones.network\_blocks.Bottleneck method), 213

forward() (easycv.models.backbones.network\_blocks.CSPConv method), 214

forward() (easycv.models.backbones.network\_blocks.DWConv method), 213

forward() (easycv.models.backbones.network\_blocks.Focus method), 215

forward() (easycv.models.backbones.network\_blocks.HSIL method), 212

forward() (easycv.models.backbones.network\_blocks.ResLayer method), 214

forward() (easycv.models.backbones.network\_blocks.SiLU method), 212

forward() (easycv.models.backbones.network\_blocks.SPPConv method), 214

forward() (easycv.models.backbones.pytorch\_image\_model\_transformer.DynamicViT method), 216

forward() (easycv.models.backbones.resnest.Bottleneck method), 217

forward() (easycv.models.backbones.resnest.GlobalAvgPool method), 217

forward() (easycv.models.backbones.resnest.ResNeSt method), 218

forward() (easycv.models.backbones.resnest.rSoftMax method), 216

forward() (easycv.models.backbones.resnest.SplAtConv2d method), 216

forward() (easycv.models.backbones.resnet.BasicBlock method), 219

forward() (easycv.models.backbones.resnet.Bottleneck method), 219

forward() (easycv.models.backbones.resnet.ResNet method), 221

forward() (easycv.models.backbones.resnet\_jit.BasicBlock method), 221

forward() (easycv.models.backbones.resnet\_jit.Bottleneck method), 222

forward() (easycv.models.backbones.resnet\_jit.ResNetJIT method), 223

forward() (easycv.models.backbones.shuffle\_transformer.Attention method), 226

forward() (easycv.models.backbones.shuffle\_transformer.Block method), 226

forward() (easycv.models.backbones.shuffle\_transformer.Mlp method), 225

forward() (easycv.models.backbones.shuffle\_transformer.PatchEmbedding method), 228

forward() (easycv.models.backbones.shuffle\_transformer.PatchMerging method), 227

forward() (easycv.models.backbones.shuffle\_transformer.ShuffleTransform method), 228

forward() (easycv.models.backbones.shuffle\_transformer.StageModule method), 227

forward() (easycv.models.backbones.swin\_transformer\_dynamic.BasicLayer method), 233

forward() (easycv.models.backbones.swin\_transformer\_dynamic.Mlp method), 229

forward() (easycv.models.backbones.swin\_transformer\_dynamic.PatchEmbedding method), 233

forward() (easycv.models.backbones.swin\_transformer\_dynamic.PatchMerge method), 232

forward() (easycv.models.backbones.swin\_transformer\_dynamic.SwinTransformer method), 235

forward() (easycv.models.backbones.swin\_transformer\_dynamic.SwinTransformer method), 231

forward() (easycv.models.backbones.swin\_transformer\_dynamic.WindowAttention method), 230

forward() (easycv.models.backbones.vit\_transformer\_dynamic.Attention method), 236

forward() (easycv.models.backbones.vit\_transformer\_dynamic.Block method), 237

forward() (easycv.models.backbones.vit\_transformer\_dynamic.DropPath method), 236

forward() (easycv.models.backbones.vit\_transformer\_dynamic.Mlp method), 236

forward() (easycv.models.backbones.vit\_transformer\_dynamic.PatchEmbedding method), 237

forward() (easycv.models.backbones.vit\_transformer\_dynamic.VisionTransformer method), 238

forward() (easycv.models.backbones.xcit\_transformer.ClassAttention method), 240

forward() (easycv.models.backbones.xcit\_transformer.ClassAttentionBlock method), 241

forward() (easycv.models.backbones.xcit\_transformer.ConvPatchEmbedding method), 239

forward() (easycv.models.backbones.xcit\_transformer.LPI method), 240

forward() (easycv.models.backbones.xcit\_transformer.PositionalEncoding method), 239

forward() (easycv.models.backbones.xcit\_transformer.XCA method), 241

forward() (easycv.models.backbones.xcit\_transformer.XCABlock method), 242

forward() (easycv.models.backbones.xcit\_transformer.XCiT method), 243

forward() (easycv.models.backbones.xcit\_transformer.XCiT method), 243

forward() (easycv.models.base.BaseModel method),



Index 339

`method`), 279  
`forward()` (`easycv.models.utils.multi_pooling.MultiAvgPooling` `method`), 280  
`forward()` (`easycv.models.utils.multi_pooling.MultiPooling` `method`), 280  
`forward()` (`easycv.models.utils.norm.IBN` `method`), 281  
`forward()` (`easycv.models.utils.norm.SyncIBN` `method`), 280  
`forward()` (`easycv.models.utils.scale.Scale` `method`), 283  
`forward()` (`easycv.models.utils.sobel.Sobel` `method`), 283  
`forward_all_selfattention()` (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformer` `method`), 235  
`forward_all_selfattention()` (`easycv.models.backbones.vit_transformer_dynamic.VisionTransformer` `method`), 238  
`forward_backbone()` (`easycv.models.classification.classification.Classification` `method`), 244  
`forward_backbone()` (`easycv.models.selfsup.moby.MoBY` `method`), 267  
`forward_backbone()` (`easycv.models.selfsup.moco.MOCO` `method`), 268  
`forward_backbone()` (`easycv.models.selfsup.simclr.SimCLR` `method`), 273  
`forward_backbone()` (`easycv.models.selfsup.swav.SWAV` `method`), 274  
`forward_compression()` (`easycv.models.detection.yolox.yolox.YOLOX` `method`), 250  
`forward_fea_and_attn()` (`easycv.models.backbones.vit_transformer_dynamic.VisionTransformer` `method`), 237  
`forward_feature()` (`easycv.models.classification.classification.Classification` `method`), 244  
`forward_feature()` (`easycv.models.selfsup.dino.DINO` `method`), 265  
`forward_feature()` (`easycv.models.selfsup.moby.MoBY` `method`), 267  
`forward_feature()` (`easycv.models.selfsup.moco.MOCO` `method`), 268  
`forward_feature()` (`easycv.models.selfsup.swav.SWAV` `method`), 274  
`forward_feature_maps()` (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformer` `method`), 235  
`forward_feature_maps()` (`easycv.models.backbones.vit_transformer_dynamic.VisionTransformer` `method`), 238  
`forward_features()` (`easycv.models.backbones.shuffle_transformer.ShuffleTransformer` `method`), 228  
`forward_features()` (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformer` `method`), 235  
`forward_features()` (`easycv.models.backbones.vit_transformer_dynamic.VisionTransformer` `method`), 238  
`forward_features()` (`easycv.models.backbones.xcit_transformer.XCiT` `method`), 243  
`forward_last_selfattention()` (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformer` `method`), 235  
`forward_last_selfattention()` (`easycv.models.backbones.vit_transformer_dynamic.VisionTransformer` `method`), 238  
`forward_loss()` (`easycv.models.selfsup.mae.MAE` `method`), 265  
`forward_return_n_last_blocks()` (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformer` `method`), 235  
`forward_return_n_last_blocks()` (`easycv.models.backbones.vit_transformer_dynamic.VisionTransformer` `method`), 238  
`forward_selfattention()` (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformer` `method`), 235  
`forward_selfattention()` (`easycv.models.backbones.vit_transformer_dynamic.VisionTransformer` `method`), 238  
`forward_test()` (`easycv.models.base.BaseModel` `method`), 284  
`forward_test()` (`easycv.models.classification.classification.Classification` `method`), 244  
`forward_test()` (`easycv.models.detection.yolox.yolox.YOLOX` `method`), 250  
`forward_test()` (`easycv.models.pose.top_down.TopDown` `method`), 261  
`forward_test()` (`easycv.models.selfsup.byol.BYOL` `method`), 263  
`forward_test()` (`easycv.models.selfsup.dino.DINO` `method`), 265  
`forward_test()` (`easycv.models.selfsup.mae.MAE` `method`), 266  
`forward_test()` (`easycv.models.selfsup.moby.MoBY` `method`), 267  
`forward_test()` (`easycv.models.selfsup.moco.MOCO` `method`), 268  
`forward_test()` (`easycv.models.selfsup.simclr.SimCLR` `method`), 273  
`forward_test()` (`easycv.models.selfsup.swav.SWAV` `method`), 274  
`forward_test_label()` (`easycv.models.classification.classification.Classification` `method`), 244  
`forward_train()` (`easycv.models.base.BaseModel` `method`), 284  
`forward_train()` (`easycv.models.classification.classification.Classification` `method`), 244  
`forward_train()` (`easycv.models.detection.yolox.yolox.YOLOX` `method`), 250

`method`), 250  
`forward_train()` (*easycv.models.pose.top\_down.TopDown* `method`), 261  
`forward_train()` (*easycv.models.selfsup.byol.BYOL* `method`), 263  
`forward_train()` (*easycv.models.selfsup.dino.DINO* `method`), 264  
`forward_train()` (*easycv.models.selfsup.mae.MAE* `method`), 265  
`forward_train()` (*easycv.models.selfsup.mixco.MIXCO* `method`), 266  
`forward_train()` (*easycv.models.selfsup.moby.MoBY* `method`), 267  
`forward_train()` (*easycv.models.selfsup.moco.MOCO* `method`), 268  
`forward_train()` (*easycv.models.selfsup.simclr.SimCLR* `method`), 273  
`forward_train()` (*easycv.models.selfsup.swav.SWAV* `method`), 274  
`forward_train_model()` (*easycv.models.selfsup.swav.SWAV* `method`), 274  
`forward_with_attention()` (*easycv.models.backbones.swin\_transformer\_dynamic.BasicLayer* `method`), 233  
`forward_with_features()` (*easycv.models.backbones.swin\_transformer\_dynamic.BasicLayer* `method`), 233  
`freeze_pretrained_layers()` (*easycv.models.backbones.swin\_transformer\_dynamic.BasicLayer* `method`), 235  
FReLU (*class in easycv.models.utils.activation*), 275  
`fuse_bn()` (*in module easycv.models.backbones.genet*), 193  
`fuseforward()` (*easycv.models.backbones.network\_blocks.BaseGnn* `method`), 213  
**G**  
`gather_tensors()` (*in module easycv.utils.gather*), 292  
`gather_tensors_batch()` (*in module easycv.utils.gather*), 292  
GatherLayer (*class in easycv.models.utils.gather\_layer*), 278  
GaussianBlur (*class in easycv.datasets.shared.pipelines.third\_transforms\_wrapper*), 119  
`gaussianBlur()` (*in module easycv.utils.preprocess\_function*), 295  
`gaussianBlurDynamic()` (*in module easycv.utils.preprocess\_function*), 295  
`gem()` (*easycv.models.utils.multi\_pooling.GeMPooling* `method`), 279  
GeMPooling (*class in easycv.models.utils.multi\_pooling*), 279  
`gen_new_list()` (*easycv.datasets.loader.DistributedGivenIterationSampler* `method`), 97  
`gen_new_list()` (*easycv.datasets.loader.sampler.DistributedGivenIterationSampler* `method`), 99  
`generate_best_metric_name()` (*in module easycv.utils.eval\_utils*), 291  
`generate_indice()` (*easycv.datasets.loader.sampler.DistributedMPSampler* `method`), 98  
`generate_new_list()` (*easycv.datasets.loader.sampler.DistributedSampler* `method`), 99  
`get()` (*easycv.core.evaluation.metric\_registry.MetricRegistry* `method`), 168  
`get()` (*easycv.utils.registry.Registry* `method`), 296  
`get_1d_sincos_pos_embed_from_grid()` (*in module easycv.models.utils.pos\_embed*), 282  
`get_2d_sincos_pos_embed()` (*in module easycv.models.utils.pos\_embed*), 282  
`get_2d_sincos_pos_embed_from_grid()` (*in module easycv.models.utils.pos\_embed*), 282  
`get_accuracy()` (*easycv.models.pose.heads.topdown\_heatmap\_base\_heads* `method`), 259  
`get_accuracy()` (*easycv.models.pose.heads.topdown\_heatmap\_simple\_heads* `method`), 260  
`get_activation()` (*in module easycv.models.backbones.network\_blocks*), 233  
`get_affine_transform()` (*in module easycv.core.post\_processing*), 174  
`get_affine_transform()` (*in module easycv.core.post\_processing.pose\_transforms*), 179  
`get_ann_info()` (*easycv.datasets.detection.data\_sources.coco.DetSourceCoco* `method`), 71  
`get_attr_info()` (*easycv.datasets.detection.data\_sources.DetSourceCoco* `method`), 68  
`get_args()` (*in module easycv.datasets.shared.pipelines.third\_transforms\_wrapper*), 118  
`get_assignments()` (*easycv.models.detection.yolox.yolo\_head.YOLOXHead* `method`), 249  
`get_cat_ids()` (*easycv.datasets.detection.data\_sources.coco.DetSourceCoco* `method`), 71  
`get_cat_ids()` (*easycv.datasets.detection.data\_sources.DetSourceCoco* `method`), 68  
`get_classifier()` (*easycv.models.backbones.shuffle\_transformer.ShuffleNet* `method`), 228  
`get_config_class_value()` (*in module easycv.utils.config\_tools*), 289  
`get_dataloader()` (*easycv.datasets.shared.dali\_tfrecord\_imagenet.DaliImNet* `method`), 127  
`get_dataloader()` (*easycv.datasets.shared.dali\_tfrecord\_multi\_view.DaliImNet* `method`), 128  
`get_dist_image()` (*in module*

`easycv.datasets.shared.odps_reader)`, 129  
`get_expansion()` (in module `easycv.models.backbones.hrnet`), 200  
`get_font_path()` (in module `easycv.core.visualization.image`), 182  
`get_in_boxes_info()` (`easycv.models.detection.yolox.yolo_head.YOLOXHead` method), 249  
`get_indexes()` (`easycv.datasets.detection.pipelines.mm_tracker.MMTracker` method), 86  
`get_indexes()` (`easycv.datasets.detection.pipelines.mm_tracker.MMTracker` method), 85  
`get_indexes()` (`easycv.datasets.detection.pipelines.MMMTracker` method), 77  
`get_indexes()` (`easycv.datasets.detection.pipelines.MMMTracker` method), 76  
`get_l1_target()` (`easycv.models.detection.yolox.yolo_head.YOLOXHead` method), 249  
`get_label_dict()` (`easycv.datasets.loader.sampler.DistributedSampler` method), 98  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 46  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 46  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 47  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 42  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 42  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 46  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 43  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 43  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 47  
`get_length()` (`easycv.datasets.classification.data_sources.ClsSource` method), 48  
`get_length()` (`easycv.datasets.detection.data_sources.cocoeval.CocoEval` method), 71  
`get_length()` (`easycv.datasets.detection.data_sources.DetectionDataset` method), 68  
`get_length()` (`easycv.datasets.pose.data_sources.PoseTopdownDataset` method), 101  
`get_length()` (`easycv.datasets.pose.data_sources.top_down_pose.TopdownPoseDataset` method), 103  
`get_length()` (`easycv.datasets.selfsup.data_sources.image_net.ImageNet` method), 110  
`get_length()` (`easycv.datasets.selfsup.data_sources.image_net.ImageNet` method), 110  
`get_length()` (`easycv.datasets.selfsup.data_sources.SSLSourceImageNet` method), 109  
`get_length()` (`easycv.datasets.selfsup.data_sources.SSLSourceImageNet` method), 110  
`get_length()` (`easycv.datasets.shared.data_sources.concat.SourceConcat` method), 114  
`get_length()` (`easycv.datasets.shared.data_sources.image_npy.ImageNpy` method), 114  
`get_length()` (`easycv.datasets.shared.data_sources.ImageNpy` method), 114  
`get_length()` (`easycv.datasets.shared.data_sources.SourceConcat` method), 114  
`get_length()` (`easycv.datasets.shared.odps_reader.OdpsReader` method), 130  
`get_length()` (`easycv.datasets.shared.OdpsReader` method), 113  
`get_loss()` (`easycv.models.pose.heads.topdown_heatmap_base_head.TopdownHeatmapBaseHead` method), 259  
`get_loss()` (`easycv.models.pose.heads.topdown_heatmap_simple_head.TopdownHeatmapSimpleHead` method), 260  
`get_loss()` (`easycv.models.detection.yolox.yolo_head.YOLOXHead` method), 249  
`get_lr()` (`easycv.models.detection.yolox_lr_hook.YOLOXLRUpdaterHook` method), 141  
`get_model_size()` (`easycv.utils.flops_counter.FlopsCounter` method), 291  
`get_model_size()` (`easycv.utils.flops_counter.FlopsCounter` method), 291  
`get_model_size()` (`easycv.utils.flops_counter.FlopsCounter` method), 291  
`get_norm_type()` (in module `easycv.hooks.sync_norm_hook`), 139  
`get_norm_type()` (in module `easycv.utils.dist_utils`), 290  
`get_output_and_grid()` (`easycv.models.detection.yolox.yolo_head.YOLOXHead` method), 249  
`get_output_type()` (`easycv.predictors.classifier.TorchClassifier` method), 144  
`get_output_type()` (`easycv.predictors.detector.TorchFaceDetector` method), 145  
`get_output_type()` (`easycv.predictors.feature_extractor.TorchFaceAttrExtractor` method), 149  
`get_output_type()` (`easycv.predictors.feature_extractor.TorchFaceFeatureExtractor` method), 147  
`get_output_type()` (`easycv.predictors.feature_extractor.TorchFeatureExtractor` method), 147  
`get_output_type()` (`easycv.predictors.feature_extractor.TorchMultiFaceDetector` method), 148  
`get_ssl_source_type()` (`easycv.predictors.interface.PredictorInterface` method), 150  
`get_ssl_source_type()` (`easycv.predictors.interface.PredictorInterfaceV2` method), 151  
`get_params_groups()` (`easycv.models.selfsup.dino.DINO` method),





attribute), 184, 185

groundtruth\_boxes (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185

groundtruth\_boxes\_absolute (*easycv.core.standard\_fields.InputDataFields* attribute), 186

groundtruth\_classes (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185

groundtruth\_difficult (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185

groundtruth\_group\_of (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185

groundtruth\_image\_classes (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185

groundtruth\_image\_classes\_num (*easycv.core.standard\_fields.InputDataFields* attribute), 185

groundtruth\_instance\_boundaries (*easycv.core.standard\_fields.InputDataFields* attribute), 185, 186

groundtruth\_instance\_classes (*easycv.core.standard\_fields.InputDataFields* attribute), 185, 186

groundtruth\_instance\_masks (*easycv.core.standard\_fields.InputDataFields* attribute), 185, 186

groundtruth\_is\_crowd (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185

groundtruth\_keypoint\_visibilities (*easycv.core.standard\_fields.InputDataFields* attribute), 185, 186

groundtruth\_keypoints (*easycv.core.standard\_fields.InputDataFields* attribute), 185, 186

groundtruth\_keypoints\_absolute (*easycv.core.standard\_fields.InputDataFields* attribute), 186

groundtruth\_label\_scores (*easycv.core.standard\_fields.InputDataFields* attribute), 185, 186

groundtruth\_label\_types (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185

groundtruth\_weights (*easycv.core.standard\_fields.InputDataFields* attribute), 185, 186

groups (*easycv.models.utils.conv\_ws.ConvWS2d* attribute), 277

GroupSampler (class in *easycv.datasets.loader*), 96

GroupSampler (class in *easycv.datasets.loader.sampler*),

## H

half\_body\_transform() (*easycv.datasets.pose.pipelines.TopDownHalfBodyTransform* static method), 104

half\_body\_transform() (*easycv.datasets.pose.pipelines.transforms.TopDownHalfBodyTransform* static method), 107

has\_batchnorms() (in module *easycv.models.selfsup.dino*), 264

height (*easycv.core.standard\_fields.InputDataFields* attribute), 185

height (*easycv.core.standard\_fields.TfExampleFields* attribute), 187, 188

HRFuseScales (class in *easycv.models.classification.necks*), 247

HRModule (class in *easycv.models.backbones.hrnet*), 201

HRNet (class in *easycv.models.backbones.hrnet*), 201

HSiLU (class in *easycv.models.backbones.network\_blocks*), 212

## I

IBN (class in *easycv.models.utils.norm*), 281

ignore\_oss\_error() (in module *easycv.file.utils*), 304

image (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185

image\_encoded (*easycv.core.standard\_fields.TfExampleFields* attribute), 187, 188

image\_format (*easycv.core.standard\_fields.TfExampleFields* attribute), 187, 188

ImageNetTFRecordPipe (class in *easycv.datasets.shared.dali\_tfrecord\_imagenet*), 127

ImageNpy (class in *easycv.datasets.shared.data\_sources*), 114

ImageNpy (class in *easycv.datasets.shared.data\_sources.image\_npy*), 114

ImageToTensor (class in *easycv.datasets.shared.pipelines.format*), 116

imshow\_bboxes() (in module *easycv.core.visualization*), 181

imshow\_bboxes() (in module *easycv.core.visualization.image*), 182

imshow\_keypoints() (in module *easycv.core.visualization*), 181

imshow\_keypoints() (in module *easycv.core.visualization.image*), 183

imshow\_label() (in module *easycv.core.visualization*), 181

imshow\_label() (in module *easycv.core.visualization.image*), 182

Inception3 (class in `easycv.models.backbones.inceptionv3`), method), 246  
 204  
 init\_weights() (`easycv.models.heads.cls_head.ClsHead` method), 250  
 inference\_model() (`easycv.models.pose.heads.topdown_heatmap_base_head.TopdownHeatmapBaseHead` method), 259  
 init\_weights() (`easycv.models.heads.latent_pred_head.LatentClsHead` method), 261  
 inference\_model() (`easycv.models.pose.heads.topdown_heatmap_simple_head.TopdownHeatmapSimpleHead` method), 261  
 init\_weights() (`easycv.models.heads.latent_pred_head.LatentPredictHead` method), 261  
 InfiniteDataLoader (class in `easycv.datasets.loader.build_loader`), 98  
 init\_weights() (`easycv.models.heads.mp_metric_head.MpMetricHead` method), 253  
 init\_before\_train() (`easycv.models.selfsup.dino.DINO` method), 254  
 init\_weights() (`easycv.models.heads.multi_cls_head.MultiClsHead` method), 254  
 init\_weights() (`easycv.models.backbones.benchmark_mlp.BenchmarkMLP` method), 191  
 init\_weights() (`easycv.models.pose.heads.topdown_heatmap_simple_head.TopdownHeatmapSimpleHead` method), 261  
 init\_weights() (`easycv.models.backbones.bninception.BNInception` method), 191  
 init\_weights() (`easycv.models.pose.top_down.TopDown` method), 261  
 init\_weights() (`easycv.models.backbones.genet.PlainNet` method), 200  
 init\_weights() (`easycv.models.selfsup.byol.BYOL` method), 263  
 init\_weights() (`easycv.models.backbones.hrnet.HRNet` method), 203  
 init\_weights() (`easycv.models.selfsup.dino.DINO` method), 264  
 init\_weights() (`easycv.models.backbones.inceptionv3.InceptionV3` method), 204  
 init\_weights() (`easycv.models.selfsup.moby.MoBY` method), 267  
 init\_weights() (`easycv.models.backbones.lighthrnet.LiteHRNet` method), 209  
 init\_weights() (`easycv.models.selfsup.moco.MOCO` method), 268  
 init\_weights() (`easycv.models.backbones.mnasnet.MNASNet` method), 211  
 init\_weights() (`easycv.models.selfsup.necks.MoBYMLP` method), 269  
 init\_weights() (`easycv.models.backbones.mobilenetv2.MobileNetV2` method), 211  
 init\_weights() (`easycv.models.selfsup.necks.NonLinearNeckSimCLR` method), 271  
 init\_weights() (`easycv.models.backbones.pytorch_image_model_zoo_v2.PytorchImageModelZooWrapper` method), 216  
 init\_weights() (`easycv.models.selfsup.necks.NonLinearNeckSwav` method), 269  
 init\_weights() (`easycv.models.backbones.resnest.ResNeSt` method), 218  
 init\_weights() (`easycv.models.selfsup.necks.NonLinearNeckV0` method), 270  
 init\_weights() (`easycv.models.backbones.resnet.ResNet` method), 221  
 init\_weights() (`easycv.models.selfsup.necks.NonLinearNeckV1` method), 270  
 init\_weights() (`easycv.models.backbones.resnet_jit.ResNetJIT` method), 223  
 init\_weights() (`easycv.models.selfsup.necks.NonLinearNeckV2` method), 271  
 init\_weights() (`easycv.models.backbones.shuffle_transformer.ShuffleTrans` method), 228  
 init\_weights() (`easycv.models.selfsup.necks.RelativeLocNeck` method), 272  
 init\_weights() (`easycv.models.backbones.swin_transformer.SwinTransformer` method), 235  
 init\_weights() (`easycv.models.selfsup.simclr.SimCLR` method), 273  
 init\_weights() (`easycv.models.backbones.vit_transformer_vit.ViT` method), 238  
 init\_weights() (`easycv.models.selfsup.swav.SWAV` method), 274  
 init\_weights() (`easycv.models.backbones.xcit_transformer.XCIT` method), 243  
 init\_weights() (`easycv.models.utils.conv_module.ConvModule` method), 276  
 init\_weights() (`easycv.models.classification.classification_utils.classification_utils_unused` method), 244  
 init\_weights() (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformerDynamic` method), 235  
 init\_weights() (`easycv.models.classification.necks.FaceIDNeck` method), 246  
 init\_yolo() (in `easycv.models.detection.yolox.yolox` module), 250  
 init\_weights() (`easycv.models.classification.necks.HRFuseScales` method), 247  
 init\_yolo() (in `easycv.models.detection.yolox.yolox` module), 251  
 init\_weights() (`easycv.models.classification.necks.LinearNeck` method), 245  
 init\_weights() (`easycv.models.detection.yolox.yolox` method), 251  
 init\_weights() (`easycv.models.classification.necks.MultiLabelNeck` method), 247  
 init\_weights() (`easycv.models.detection.yolox.yolo_head.YOLOXHead` method), 248  
 init\_weights() (`easycv.models.classification.necks.ReTrivalNeck` method), 248

|                            |                                                                                                        |                                                                                                                       |
|----------------------------|--------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| InputDataFields            | (class in <code>easycore.core.standard_fields</code> ), 184                                            | <code>is_instance_from_str()</code> (static method), 196                                                              |
| instance_boundaries        | ( <code>easycore.core.standard_fields.TfExampleFields</code> attribute), 188, 189                      | ( <code>easycore.models.backbones.genet.PlainNetBasicBlockClass</code> static method), 193                            |
| instance_classes           | ( <code>easycore.core.standard_fields.TfExampleFields</code> attribute), 188, 189                      | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.RELU</code> static method), 197           |
| instance_masks             | ( <code>easycore.core.standard_fields.TfExampleFields</code> attribute), 188, 189                      | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.ResBlock</code> static method), 197       |
| interpolate_pos_embed()    | (in module <code>easycore.models.utils.pos_embed</code> ), 282                                         | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.Sequential</code> static method), 197     |
| interpolate_pos_encoding() | ( <code>easycore.models.backbones.vit_transformer_dynamic.VisionTransformerDynamic</code> method), 238 | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.SuperResK1DW</code> static method), 199   |
| interval                   | ( <code>easycore.hooks.eval_hook.DistEvalHook</code> attribute), 136                                   | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.SuperResK1DWK1</code> static method), 199 |
| interval                   | ( <code>easycore.hooks.eval_hook.EvalHook</code> attribute), 135                                       | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.SuperResK1KXK1</code> static method), 199 |
| Invert                     | (class in <code>easycore.datasets.classification.pipelines.auto_augment</code> ), 60                   | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.SuperResK1KXK1</code> static method), 198 |
| IO                         | (class in <code>easycore.file.file_io</code> ), 300                                                    | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.SuperResK1KXK1</code> static method), 199 |
| IOBase                     | (class in <code>easycore.file.base</code> ), 299                                                       | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.SuperResK1KXK1</code> static method), 198 |
| IOLocal                    | (class in <code>easycore.file.base</code> ), 300                                                       | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.SuperResK1KXK1</code> static method), 199 |
| IOUloss                    | (class in <code>easycore.models.loss.iou_loss</code> ), 255                                            | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.SuperResK1KXK1</code> static method), 198 |
| is_child_of()              | (in module <code>easycore.datasets.shared.pipelines.third_transforms</code> ), 118                     | <code>is_instance_from_str()</code> ( <code>easycore.models.backbones.genet.SuperResK1KXK1</code> static method), 198 |
| is_dali_dataset_type()     | (in module <code>easycore.datasets.utils.type_util</code> ), 131                                       | <code>is_itag_v2()</code> (in module <code>easycore.datasets.detection.data_sources.pai_format</code> ), 72           |
| is_distributed()           | (in module <code>easycore.models.utils.dist_utils</code> ), 278                                        | <code>is_master()</code> (in module <code>easycore.utils.dist_utils</code> ), 290                                     |
| is_instance_from_str()     | ( <code>easycore.models.backbones.genet.AdaptiveAvgPool</code> static method), 194                     | <code>is_oss_path()</code> (in module <code>easycore.file.utils</code> ), 304                                         |
| is_instance_from_str()     | ( <code>easycore.models.backbones.genet.BN</code> static method), 194                                  | <code>is_parallel()</code> (in module <code>easycore.utils.dist_utils</code> ), 290                                   |
| is_instance_from_str()     | ( <code>easycore.models.backbones.genet.ConvDW</code> static method), 194                              | <code>is_port_used()</code> (in module <code>easycore.utils.test_util</code> ), 296                                   |
| is_instance_from_str()     | ( <code>easycore.models.backbones.genet.ConvKX</code> static method), 195                              | <code>is_supported_instance()</code> (in module <code>easycore.utils.flops_counter</code> ), 292                      |
| is_instance_from_str()     | ( <code>easycore.models.backbones.genet.Flatten</code> static method), 195                             | <code>is_writable()</code> ( <code>easycore.file.base.IOBase</code> method), 299                                      |
| is_instance_from_str()     | ( <code>easycore.models.backbones.genet.Linear</code> static method), 195                              | <code>isdir()</code> ( <code>easycore.file.base.IOBase</code> method), 299                                            |
| is_instance_from_str()     | ( <code>easycore.models.backbones.genet.MaxPool</code> static method), 196                             | <code>isdir()</code> ( <code>easycore.file.base.IOLocal</code> method), 300                                           |
| is_instance_from_str()     | ( <code>easycore.models.backbones.genet.MultiSumBlock</code> static method), 196                       | <code>isdir()</code> ( <code>easycore.file.file_io.IO</code> method), 303                                             |
|                            |                                                                                                        | <code>isfile()</code> ( <code>easycore.file.base.IOBase</code> method), 299                                           |
|                            |                                                                                                        | <code>isfile()</code> ( <code>easycore.file.base.IOLocal</code> method), 300                                          |
|                            |                                                                                                        | <code>isfile()</code> ( <code>easycore.file.file_io.IO</code> method), 303                                            |
|                            |                                                                                                        | <code>islocal()</code> ( <code>easycore.file.base.IOBase</code> method), 299                                          |
|                            |                                                                                                        | <code>IterativeHead</code> (class in <code>easycore.models.backbones.lighthnet</code> ), 207                          |
|                            |                                                                                                        | <code>iterencode()</code> ( <code>easycore.utils.json_utils.MyEncoder</code> method), 293                             |
|                            |                                                                                                        | <b>J</b>                                                                                                              |
|                            |                                                                                                        | <code>JointsMSELoss</code> (class in <code>easycore.models.loss.mse_loss</code> ), 255                                |



## K

`kernel_size` (*easycv.models.utils.conv\_ws.ConvWS2d* attribute), 277

`key` (*easycv.core.standard\_fields.DetectionResultFields* attribute), 186

`key` (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185

`keypoint_pck_accuracy()` (in module *easycv.core.evaluation.top\_down\_eval*), 169

`keypoints_from_heatmaps()` (in module *easycv.core.evaluation.top\_down\_eval*), 170

## L

`label_map` (*easycv.core.standard\_fields.InputDataFields* attribute), 186

`Lambda` (class in *easycv.datasets.shared.pipelines.third\_transforms\_wrapper*), 119

`LARS` (class in *easycv.core.optimizer.lars*), 172

`last_modified()` (*easycv.file.base.IOBase* method), 299

`last_modified()` (*easycv.file.base.IOLocal* method), 300

`last_modified()` (*easycv.file.file\_io.IO* method), 304

`last_modified_str()` (*easycv.file.base.IOBase* method), 299

`last_modified_str()` (*easycv.file.base.IOLocal* method), 300

`last_modified_str()` (*easycv.file.file\_io.IO* method), 304

`LatentClsHead` (class in *easycv.models.heads.latent\_pred\_head*), 253

`LatentPredictHead` (class in *easycv.models.heads.latent\_pred\_head*), 253

`Lighting` (class in *easycv.datasets.selfsup.pipelines*), 111

`Lighting` (class in *easycv.datasets.selfsup.pipelines.transforms*), 111

`Linear` (class in *easycv.models.backbones.genet*), 195

`linear_flops_counter_hook()` (in module *easycv.utils.flops\_counter*), 292

`LinearNeck` (class in *easycv.models.classification.necks*), 245

`LinearTransformation` (class in *easycv.datasets.shared.pipelines.third\_transforms\_wrapper*), 119

`listdir()` (*easycv.file.base.IOBase* method), 299

`listdir()` (*easycv.file.base.IOLocal* method), 300

`listdir()` (*easycv.file.file\_io.IO* method), 302

`LiteHRModule` (class in *easycv.models.backbones.lightrnet*), 208

`LiteHRNet` (class in *easycv.models.backbones.lightrnet*), 208

`load_annotations()` (*easycv.datasets.classification.data\_sources.ClsSource* method), 46

`load_annotations()` (*easycv.datasets.detection.data\_sources.coco.DetectionSource* method), 71

`load_annotations()` (*easycv.datasets.detection.data\_sources.DetectionSource* method), 68

`load_checkpoint()` (*easycv.runner.ev\_runner.EVRunner* method), 306

`load_checkpoint()` (in module *easycv.utils.checkpoint*), 288

`load_image()` (*easycv.datasets.pose.data\_sources.PoseTopDownSource* method), 101

`load_image()` (*easycv.datasets.pose.data\_sources.top\_down.PoseTopDownSource* method), 103

`LoadAnnotations` (class in *easycv.datasets.detection.pipelines*), 82

`LoadAnnotations` (class in *easycv.datasets.detection.pipelines.mm\_transforms*), 91

`LoadAnnotations()` (*easycv.core.evaluation.coco\_tools.COCOWrapper* method), 161

`LoadImage` (class in *easycv.predictors.pose\_predictor*), 152

`LoadImageFromFile` (class in *easycv.datasets.detection.pipelines*), 81

`LoadImageFromFile` (class in *easycv.datasets.detection.pipelines.mm\_transforms*), 91

`LoadMultiChannelImageFromFiles` (class in *easycv.datasets.detection.pipelines*), 82

`LoadMultiChannelImageFromFiles` (class in *easycv.datasets.detection.pipelines.mm\_transforms*), 91

`local_rank()` (in module *easycv.utils.dist\_utils*), 290

`log()` (*easycv.hooks.tensorboard.TensorboardLoggerHookV2* method), 140

`log()` (*easycv.hooks.wandb.WandbLoggerHookV2* method), 140

`logits()` (*easycv.models.backbones.bninception.BNInception* method), 191

`loss()` (*easycv.models.heads.cls\_head.ClsHead* method), 252

`loss()` (*easycv.models.heads.mp\_metric\_head.MpMetricHead* method), 254

`loss()` (*easycv.models.heads.multi\_cls\_head.MultiClsHead* method), 255

`LpDistance()` (in module *easycv.utils.metric\_distance*), 294

`LPI` (class in *easycv.models.backbones.xcit\_transformer*), 240

## M

`MAE` (class in *easycv.models.selfsup.mae*), 265

`MAEFtAugment` (class in *easycv.models.selfsup.mae*), 265

[easycv.datasets.selfsup.pipelines.transforms](#)), 111  
[MAENeck](#) (class in [easycv.models.selfsup.necks](#)), 272  
[make\\_group\\_layer\(\)](#) ([easycv.models.backbones.darknet.Darknet](#) method), 192  
[make\\_res\\_layer\(\)](#) (in module [easycv.models.backbones.resnet](#)), 219  
[make\\_res\\_layer\(\)](#) (in module [easycv.models.backbones.resnet\\_jit](#)), 222  
[make\\_res\\_layer\(\)](#) (in module [easycv.models.backbones.resnext](#)), 224  
[make\\_spp\\_block\(\)](#) ([easycv.models.backbones.darknet.Darknet](#) method), 192  
[makedirs\(\)](#) ([easycv.file.base.IOBase](#) method), 299  
[makedirs\(\)](#) ([easycv.file.base.IOLocal](#) method), 300  
[makedirs\(\)](#) ([easycv.file.file\\_io.IO](#) method), 303  
[makeplot\(\)](#) ([easycv.core.evaluation.custom\\_cocotools.cocoeval.COCEval](#) method), 156  
[mask](#) ([easycv.core.standard\\_fields.InputDataFields](#) attribute), 185  
[MaskedAutoencoderViT](#) (class in [easycv.models.backbones.mae\\_vit\\_transformer](#)), 210  
[MaxPool](#) (class in [easycv.models.backbones.genet](#)), 196  
[md5\(\)](#) ([easycv.file.base.IOBase](#) method), 299  
[merge\\_hparams\(\)](#) (in module [easycv.datasets.classification.pipelines.auto\\_augment](#)), 54  
[metric\\_names](#) ([easycv.core.evaluation.base\\_evaluator.Evaluator](#) property), 157  
[MetricRegistry](#) (class in [easycv.core.evaluation.metric\\_registry](#)), 168  
[MIXCO](#) (class in [easycv.models.selfsup.mixco](#)), 266  
[mixUp\(\)](#) (in module [easycv.utils.preprocess\\_function](#)), 295  
[mixup\\_loss\(\)](#) ([easycv.models.heads.cls\\_head.ClsHead](#) method), 252  
[mixUpCls\(\)](#) (in module [easycv.utils.preprocess\\_function](#)), 295  
[Mlp](#) (class in [easycv.models.backbones.shuffle\\_transformer](#)), 225  
[Mlp](#) (class in [easycv.models.backbones.swin\\_transformer\\_dynamic](#)), 229  
[Mlp](#) (class in [easycv.models.backbones.vit\\_transformer\\_dynamic](#)), 236  
[MMAutoAugment](#) (class in [easycv.datasets.classification.pipelines](#)), 50  
[MMAutoAugment](#) (class in [easycv.datasets.classification.pipelines.auto\\_augment](#)), 54  
[mmcv\\_config\\_fromfile\(\)](#) (in module [easycv.utils.config\\_tools](#)), 289  
[mmcv\\_file2dict\\_base\(\)](#) (in module [easycv.utils.config\\_tools](#)), 289  
[mmcv\\_file2dict\\_raw\(\)](#) (in module [easycv.utils.config\\_tools](#)), 289  
[MMMixUp](#) (class in [easycv.datasets.detection.pipelines](#)), 76  
[MMMixUp](#) (class in [easycv.datasets.detection.pipelines.mm\\_transforms](#)), 85  
[MMMosaic](#) (class in [easycv.datasets.detection.pipelines](#)), 75  
[MMMosaic](#) (class in [easycv.datasets.detection.pipelines.mm\\_transforms](#)), 84  
[MMMMultiScaleFlipAug](#) (class in [easycv.datasets.detection.pipelines](#)), 83  
[MMMMultiScaleFlipAug](#) (class in [easycv.datasets.detection.pipelines.mm\\_transforms](#)), 92  
[MMNormalize](#) (class in [easycv.datasets.detection.pipelines](#)), 81  
[MMNormalize](#) (class in [easycv.datasets.detection.pipelines.mm\\_transforms](#)), 90  
[MMPad](#) (class in [easycv.datasets.detection.pipelines](#)), 81  
[MMPad](#) (class in [easycv.datasets.detection.pipelines.mm\\_transforms](#)), 90  
[MMPhotoMetricDistortion](#) (class in [easycv.datasets.detection.pipelines](#)), 78  
[MMPhotoMetricDistortion](#) (class in [easycv.datasets.detection.pipelines.mm\\_transforms](#)), 87  
[MMRandAugment](#) (class in [easycv.datasets.classification.pipelines](#)), 51  
[MMRandAugment](#) (class in [easycv.datasets.classification.pipelines.auto\\_augment](#)), 56  
[MMRandomAffine](#) (class in [easycv.datasets.detection.pipelines](#)), 77  
[MMRandomAffine](#) (class in [easycv.datasets.detection.pipelines.mm\\_transforms](#)), 86  
[MMRandomErasing](#) (class in [easycv.datasets.classification.pipelines](#)), 63  
[MMRandomErasing](#) (class in [easycv.datasets.classification.pipelines.transform](#)), 63  
[MMRandomFlip](#) (class in [easycv.datasets.detection.pipelines](#)), 80  
[MMRandomFlip](#) (class in [easycv.datasets.detection.pipelines.mm\\_transforms](#)), 89  
[MMResize](#) (class in [easycv.datasets.detection.pipelines](#)), 79  
[MMResize](#) (class in [easycv.datasets.detection.pipelines.mm\\_transforms](#)), 88

**MMToTensor** (class in *easy cv.datasets.detection.pipelines*), 75  
**MMToTensor** (class in *easy cv.datasets.detection.pipelines.mm\_transforms*), 84  
**MNASNet** (class in *easy cv.models.backbones.mnasnet*), 211  
**MobileNetV2** (class in *easy cv.models.backbones.mobilenetv2*), 211  
**MoBY** (class in *easy cv.models.selfsup.moby*), 266  
**MoBYMLP** (class in *easy cv.models.selfsup.necks*), 269  
**MOCO** (class in *easy cv.models.selfsup.moco*), 268  
**mode** (*easy cv.hooks.eval\_hook.DistEvalHook* attribute), 136  
**mode** (*easy cv.hooks.eval\_hook.EvalHook* attribute), 135  
**ModelEMA** (class in *easy cv.hooks.ema\_hook*), 134  
**ModelParallelAMSoftmaxLoss** (class in *easy cv.models.loss.pytorch\_metric\_learning*), 258  
**ModelParallelSoftmaxLoss** (class in *easy cv.models.loss.pytorch\_metric\_learning*), 257  
**module**  
     *easy cv*, 299  
     *easy cv.apis*, 37  
     *easy cv.apis.export*, 37  
     *easy cv.apis.test*, 37  
     *easy cv.apis.train*, 38  
     *easy cv.apis.train\_misc*, 39  
     *easy cv.core*, 155  
     *easy cv.core.evaluation*, 155  
     *easy cv.core.evaluation.ap*, 156  
     *easy cv.core.evaluation.auc\_eval*, 157  
     *easy cv.core.evaluation.base\_evaluator*, 157  
     *easy cv.core.evaluation.builder*, 157  
     *easy cv.core.evaluation.classification\_eval*, 157  
     *easy cv.core.evaluation.coco\_evaluation*, 158  
     *easy cv.core.evaluation.coco\_tools*, 160  
     *easy cv.core.evaluation.custom\_cocotools*, 155  
     *easy cv.core.evaluation.custom\_cocotools.cocoeval*, 155  
     *easy cv.core.evaluation.faceid\_pair\_eval*, 167  
     *easy cv.core.evaluation.metric\_registry*, 168  
     *easy cv.core.evaluation.mse\_eval*, 168  
     *easy cv.core.evaluation.retrival\_topk\_eval*, 169  
     *easy cv.core.evaluation.top\_down\_eval*, 169  
     *easy cv.core.optimizer*, 172  
     *easy cv.core.optimizer.lars*, 172  
     *easy cv.core.optimizer.ranger*, 173  
     *easy cv.core.post\_processing*, 173  
     *easy cv.core.post\_processing.nms*, 177  
     *easy cv.core.post\_processing.pose\_transforms*, 178  
     *easy cv.core.standard\_fields*, 184  
     *easy cv.core.visualization*, 181  
     *easy cv.core.visualization.image*, 182  
     *easy cv.datasets*, 41  
     *easy cv.datasets.builder*, 131  
     *easy cv.datasets.classification*, 41  
     *easy cv.datasets.classification.data\_sources*, 42  
     *easy cv.datasets.classification.data\_sources.cifar*, 46  
     *easy cv.datasets.classification.data\_sources.class\_list*, 46  
     *easy cv.datasets.classification.data\_sources.fashiongen*, 47  
     *easy cv.datasets.classification.data\_sources.image\_list*, 47  
     *easy cv.datasets.classification.data\_sources.imagenet\_t*, 48  
     *easy cv.datasets.classification.data\_sources.utils*, 48  
     *easy cv.datasets.classification.odps*, 64  
     *easy cv.datasets.classification.pipelines*, 50  
     *easy cv.datasets.classification.pipelines.auto\_augment*, 54  
     *easy cv.datasets.classification.pipelines.transform*, 63  
     *easy cv.datasets.classification.raw*, 64  
     *easy cv.datasets.detection*, 65  
     *easy cv.datasets.detection.data\_sources*, 67  
     *easy cv.datasets.detection.data\_sources.coco*, 71  
     *easy cv.datasets.detection.data\_sources.pai\_format*, 72  
     *easy cv.datasets.detection.data\_sources.raw*, 73  
     *easy cv.datasets.detection.data\_sources.utils*, 74  
     *easy cv.datasets.detection.data\_sources.voc*, 74  
     *easy cv.datasets.detection.mix*, 93  
     *easy cv.datasets.detection.pipelines*, 75  
     *easy cv.datasets.detection.pipelines.mm\_transforms*, 84  
     *easy cv.datasets.detection.raw*, 95  
     *easy cv.datasets.loader*, 96  
     *easy cv.datasets.loader.build\_loader*, 97  
     *easy cv.datasets.loader.sampler*, 98

easycv.datasets.pose, 100  
 easycv.datasets.pose.data\_sources, 100  
 easycv.datasets.pose.data\_sources.coco, 102  
 easycv.datasets.pose.data\_sources.top\_down, 103  
 easycv.datasets.pose.pipelines, 103  
 easycv.datasets.pose.pipelines.transforms, 106  
 easycv.datasets.pose.top\_down, 109  
 easycv.datasets.registry, 131  
 easycv.datasets.selfsup, 109  
 easycv.datasets.selfsup.data\_sources, 109  
 easycv.datasets.selfsup.data\_sources.image\_list, 110  
 easycv.datasets.selfsup.data\_sources.imagenet\_finetune, 110  
 easycv.datasets.selfsup.pipelines, 111  
 easycv.datasets.selfsup.pipelines.transforms, 111  
 easycv.datasets.shared, 112  
 easycv.datasets.shared.base, 126  
 easycv.datasets.shared.dali\_tfrecord\_imagenet, 127  
 easycv.datasets.shared.dali\_tfrecord\_multi\_view, 128  
 easycv.datasets.shared.data\_sources, 114  
 easycv.datasets.shared.data\_sources.concat, 114  
 easycv.datasets.shared.data\_sources.image\_npy, 114  
 easycv.datasets.shared.dataset\_wrappers, 129  
 easycv.datasets.shared.multi\_view, 129  
 easycv.datasets.shared.odps\_reader, 129  
 easycv.datasets.shared.pipelines, 115  
 easycv.datasets.shared.pipelines.dali\_transformer, 115  
 easycv.datasets.shared.pipelines.format, 116  
 easycv.datasets.shared.pipelines.third\_transformer\_wrapper, 118  
 easycv.datasets.shared.pipelines.transforms, 126  
 easycv.datasets.shared.raw, 130  
 easycv.datasets.utils, 130  
 easycv.datasets.utils.tfrecord\_util, 130  
 easycv.datasets.utils.type\_util, 131  
 easycv.file, 299  
 easycv.file.base, 299  
 easycv.file.file\_io, 300  
 easycv.file.utils, 304  
 easycv.hooks, 133  
 easycv.hooks.best\_ckpt\_saver\_hook, 133  
 easycv.hooks.builder, 133  
 easycv.hooks.byol\_hook, 134  
 easycv.hooks.dino\_hook, 134  
 easycv.hooks.ema\_hook, 134  
 easycv.hooks.eval\_hook, 135  
 easycv.hooks.export\_hook, 136  
 easycv.hooks.extractor, 137  
 easycv.hooks.optimizer\_hook, 137  
 easycv.hooks.oss\_sync\_hook, 137  
 easycv.hooks.registry, 138  
 easycv.hooks.show\_time\_hook, 138  
 easycv.hooks.swav\_hook, 138  
 easycv.hooks.sync\_norm\_hook, 139  
 easycv.hooks.sync\_random\_size\_hook, 139  
 easycv.hooks.tensorboard, 140  
 easycv.hooks.fastai\_hooks, 140  
 easycv.hooks.wandb, 140  
 easycv.hooks.yolox\_lr\_hook, 140  
 easycv.hooks.yolox\_mode\_switch\_hook, 141  
 easycv.models, 191  
 easycv.models.backbones, 191  
 easycv.models.backbones.benchmark\_mlp, 191  
 easycv.models.backbones.bninception, 191  
 easycv.models.backbones.darknet, 192  
 easycv.models.backbones.genet, 193  
 easycv.models.backbones.hrnet, 200  
 easycv.models.backbones.inceptionv3, 204  
 easycv.models.backbones.lightrnet, 204  
 easycv.models.backbones.mae\_vit\_transformer, 210  
 easycv.models.backbones.mnasnet, 211  
 easycv.models.backbones.mobilenetv2, 211  
 easycv.models.backbones.network\_blocks, 212  
 easycv.models.backbones.pytorch\_image\_models\_wrapper, 215  
 easycv.models.backbones.resnest, 216  
 easycv.models.backbones.resnet, 219  
 easycv.models.backbones.resnet\_jit, 221  
 easycv.models.backbones.resnext, 224  
 easycv.models.backbones.shuffle\_transformer, 225  
 easycv.models.backbones.swin\_transformer\_dynamic, 229  
 easycv.models.backbones.vit\_transformer\_dynamic, 236  
 easycv.models.backbones.xcit\_transformer, 239  
 easycv.models.base, 284  
 easycv.models.builder, 285  
 easycv.models.classification, 244  
 easycv.models.classification.classification, 244  
 easycv.models.classification.necks, 245

[easycv.models.detection](#), 248  
[easycv.models.detection.utils](#), 248  
[easycv.models.detection.utils.bboxes](#), 248  
[easycv.models.detection.yolox](#), 248  
[easycv.models.detection.yolox.yolo\\_head](#), 248  
[easycv.models.detection.yolox.yolo\\_pafpn](#), 249  
[easycv.models.detection.yolox.yolox](#), 250  
[easycv.models.detection.yolox\\_edge](#), 251  
[easycv.models.detection.yolox\\_edge.yolox\\_edge](#), 251  
[easycv.models.heads](#), 251  
[easycv.models.heads.cls\\_head](#), 251  
[easycv.models.heads.contrastive\\_head](#), 252  
[easycv.models.heads.latent\\_pred\\_head](#), 253  
[easycv.models.heads.mp\\_metric\\_head](#), 253  
[easycv.models.heads.multi\\_cls\\_head](#), 254  
[easycv.models.loss](#), 255  
[easycv.models.loss.iou\\_loss](#), 255  
[easycv.models.loss.mse\\_loss](#), 255  
[easycv.models.loss.pytorch\\_metric\\_learning](#), 256  
[easycv.models.modelzoo](#), 285  
[easycv.models.pose](#), 258  
[easycv.models.pose.heads](#), 258  
[easycv.models.pose.heads.topdown\\_heatmap\\_base\\_head](#), 259  
[easycv.models.pose.heads.topdown\\_heatmap\\_simple\\_head](#), 259  
[easycv.models.pose.top\\_down](#), 261  
[easycv.models.registry](#), 285  
[easycv.models.selfsup](#), 263  
[easycv.models.selfsup.byol](#), 263  
[easycv.models.selfsup.dino](#), 263  
[easycv.models.selfsup.mae](#), 265  
[easycv.models.selfsup.mixco](#), 266  
[easycv.models.selfsup.moby](#), 266  
[easycv.models.selfsup.moco](#), 268  
[easycv.models.selfsup.necks](#), 269  
[easycv.models.selfsup.simclr](#), 273  
[easycv.models.selfsup.swav](#), 274  
[easycv.models.utils](#), 275  
[easycv.models.utils.accuracy](#), 275  
[easycv.models.utils.activation](#), 275  
[easycv.models.utils.conv\\_module](#), 276  
[easycv.models.utils.conv\\_ws](#), 277  
[easycv.models.utils.dist\\_utils](#), 278  
[easycv.models.utils.gather\\_layer](#), 278  
[easycv.models.utils.init\\_weights](#), 279  
[easycv.models.utils.multi\\_pooling](#), 279  
[easycv.models.utils.norm](#), 280  
[easycv.models.utils.ops](#), 282  
[easycv.models.utils.pos\\_embed](#), 282  
[easycv.models.utils.res\\_layer](#), 282  
[easycv.models.utils.scale](#), 283  
[easycv.models.utils.sobel](#), 283  
[easycv.predictors](#), 143  
[easycv.predictors.base](#), 143  
[easycv.predictors.builder](#), 143  
[easycv.predictors.classifier](#), 144  
[easycv.predictors.detector](#), 145  
[easycv.predictors.feature\\_extractor](#), 147  
[easycv.predictors.interface](#), 150  
[easycv.predictors.pose\\_predictor](#), 152  
[easycv.runner](#), 305  
[easycv.runner.ev\\_runner](#), 305  
[easycv.toolkit](#), 306  
[easycv.toolkit.prune](#), 306  
[easycv.toolkit.quantize](#), 306  
[easycv.toolkit.quantize.quantize\\_utils](#), 306  
[easycv.utils](#), 287  
[easycv.utils.alias\\_multinomial](#), 287  
[easycv.utils.bbox\\_util](#), 287  
[easycv.utils.checkpoint](#), 288  
[easycv.utils.collect](#), 289  
[easycv.utils.collect\\_env](#), 289  
[easycv.utils.config\\_tools](#), 289  
[easycv.utils.constant](#), 290  
[easycv.utils.dist\\_utils](#), 290  
[easycv.utils.eval\\_utils](#), 291  
[easycv.utils.flops\\_counter](#), 291  
[easycv.utils.gather](#), 292  
[easycv.utils.json\\_utils](#), 292  
[easycv.utils.logger](#), 294  
[easycv.utils.metric\\_distance](#), 294  
[easycv.utils.misc](#), 295  
[easycv.utils.preprocess\\_function](#), 295  
[easycv.utils.profiling](#), 295  
[easycv.utils.py\\_util](#), 296  
[easycv.utils.registry](#), 296  
[easycv.utils.test\\_util](#), 296  
[easycv.utils.user\\_config\\_params\\_utils](#), 297  
[easycv.version](#), 307  
[module\\_dict](#) (*easycv.utils.registry.Registry* property), 296  
[momentum\\_update\\_key\\_encoder\(\)](#) (*easycv.models.selfsup.dino.DINO* method), 264  
[move\(\)](#) (*easycv.file.base.IOBase* method), 299  
[move\(\)](#) (*easycv.file.base.IOLocal* method), 300  
[move\(\)](#) (*easycv.file.file\_io.IO* method), 301  
[MpMetrixHead](#) (class in *easycv.models.heads.mp\_metric\_head*), 253  
[MSEEvaluator](#) (class in *easycv.core.evaluation.mse\_eval*), 168



- [multi\\_apply\(\)](#) (in module `easycv.utils.misc`), 295  
[multi\\_gpu\\_test\(\)](#) (in module `easycv.apis.test`), 37  
[MultiAvgPooling](#) (class `easycv.models.utils.multi_pooling`), 280  
[MultiClsHead](#) (class `easycv.models.heads.multi_cls_head`), 254  
[MultiCropWrapper](#) (class `easycv.models.selfsup.dino`), 263  
[MultiLinearNeck](#) (class `easycv.models.classification.necks`), 246  
[MultiPooling](#) (class `easycv.models.utils.multi_pooling`), 279  
[MultiPrototypes](#) (class `easycv.models.selfsup.swav`), 274  
[MultiSumBlock](#) (class `easycv.models.backbones.genet`), 196  
[MultiViewDataset](#) (class in `easycv.datasets.shared`), 113  
[MultiViewDataset](#) (class `easycv.datasets.shared.multi_view`), 129  
[mute\\_stderr\(\)](#) (in module `easycv.file.utils`), 304  
[MyEncoder](#) (class in `easycv.utils.json_utils`), 292
- ## N
- [name](#) (`easycv.utils.registry.Registry` property), 296  
[no\\_weight\\_decay\(\)](#) (`easycv.models.backbones.shuffle_transformer.ShuffleTransformer` method), 228  
[no\\_weight\\_decay\(\)](#) (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformer` method), 235  
[no\\_weight\\_decay\(\)](#) (`easycv.models.backbones.xcit_transformer.XCARTy`), 201  
[no\\_weight\\_decay\(\)](#) (`easycv.models.backbones.xcit_transformer.XCFTy`), 219  
[no\\_weight\\_decay\(\)](#) (`easycv.models.backbones.xcit_transformer.XCFTy` method), 243  
[no\\_weight\\_decay\\_keywords\(\)](#) (`easycv.models.backbones.shuffle_transformer.ShuffleTransformer` method), 228  
[no\\_weight\\_decay\\_keywords\(\)](#) (`easycv.models.backbones.swin_transformer_dynamic.SwinTransformer` method), 235  
[nondist\\_forward\\_collect\(\)](#) (in module `easycv.utils.collect`), 289  
[NonLinearNeckSimCLR](#) (class `easycv.models.selfsup.necks`), 271  
[NonLinearNeckSwav](#) (class `easycv.models.selfsup.necks`), 269  
[NonLinearNeckV0](#) (class `easycv.models.selfsup.necks`), 270  
[NonLinearNeckV1](#) (class `easycv.models.selfsup.necks`), 270  
[NonLinearNeckV2](#) (class `easycv.models.selfsup.necks`), 270  
[norm](#) (`easycv.models.utils.conv_module.ConvModule` property), 276  
[norm1](#) (`easycv.models.backbones.hrnet.Bottleneck` property), 201  
[in norm1](#) (`easycv.models.backbones.hrnet.HRNet` property), 203  
[in norm1](#) (`easycv.models.backbones.resnet.BasicBlock` property), 219  
[in norm1](#) (`easycv.models.backbones.resnet.Bottleneck` property), 219  
[in norm1](#) (`easycv.models.backbones.resnet.ResNet` property), 221  
[in norm1](#) (`easycv.models.backbones.resnet_jit.BasicBlock` property), 221  
[in norm1](#) (`easycv.models.backbones.resnet_jit.Bottleneck` property), 222  
[in norm1](#) (`easycv.models.backbones.resnet_jit.ResNetJIT` property), 223  
[norm2](#) (`easycv.models.backbones.hrnet.Bottleneck` property), 201  
[in norm2](#) (`easycv.models.backbones.hrnet.HRNet` property), 203  
[norm2](#) (`easycv.models.backbones.resnet.BasicBlock` property), 219  
[norm2](#) (`easycv.models.backbones.resnet.Bottleneck` property), 219  
[norm2](#) (`easycv.models.backbones.resnet_jit.BasicBlock` property), 221  
[norm2](#) (`easycv.models.backbones.resnet_jit.Bottleneck` property), 222  
[norm3](#) (`easycv.models.backbones.hrnet.Bottleneck` property), 201  
[norm3](#) (`easycv.models.backbones.resnet.BasicBlock` property), 219  
[norm3](#) (`easycv.models.backbones.resnet.Bottleneck` property), 219  
[norm3](#) (`easycv.models.backbones.resnet_jit.BasicBlock` property), 221  
[norm3](#) (`easycv.models.backbones.resnet_jit.Bottleneck` property), 222  
[Normalize](#) (class in `easycv.datasets.shared.pipelines.third_transforms_wrapper`), 120  
[NormalizeTensor](#) (class `easycv.datasets.detection.pipelines`), 75  
[NormalizeTensor](#) (class `easycv.datasets.detection.pipelines.mm_transforms`), 84  
[NullContextWrapper](#) (class in `easycv.file.file_io`), 304  
[num\\_detections](#) (`easycv.core.standard_fields.DetectionResultFields` attribute), 186, 187  
[num\\_groundtruth\\_boxes](#) (`easycv.core.standard_fields.InputDataFields` attribute), 185, 186  
[num\\_workers](#) (`easycv.datasets.loader.build_loader.InfiniteDataLoader` attribute), 98  
[NumpyToPIL](#) (class in `easycv.predictors.base`), 143
- ## O
- [obj2tensor\(\)](#) (in module `easycv.utils.dist_utils`), 290

`object_bbox_xmax` (`easycv.core.standard_fields.TfExampleFields` attribute), 187, 189  
`object_bbox_xmin` (`easycv.core.standard_fields.TfExampleFields` attribute), 187, 189  
`object_bbox_ymax` (`easycv.core.standard_fields.TfExampleFields` attribute), 187, 189  
`object_bbox_ymin` (`easycv.core.standard_fields.TfExampleFields` attribute), 187, 189  
`object_class_label` (`easycv.core.standard_fields.TfExampleFields` attribute), 187, 189  
`object_class_text` (`easycv.core.standard_fields.TfExampleFields` attribute), 187, 189  
`object_depiction` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
`object_difficult` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
`object_group_of` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
`object_is_crowd` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
`object_occluded` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
`object_segment_area` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
`object_truncated` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
`object_view` (`easycv.core.standard_fields.TfExampleFields` attribute), 187, 189  
`object_weight` (`easycv.core.standard_fields.TfExampleFields` attribute), 188, 189  
`OdpsReader` (class in `easycv.datasets.shared`), 112  
`OdpsReader` (class in `easycv.datasets.shared.odps_reader`), 129  
`oks_iou()` (in module `easycv.core.post_processing.nms`), 177  
`oks_nms()` (in module `easycv.core.post_processing`), 176  
`oks_nms()` (in module `easycv.core.post_processing.nms`), 177  
`open()` (`easycv.file.base.IOBase` method), 299  
`open()` (`easycv.file.base.IOLocal` method), 300  
`open()` (`easycv.file.file_io.IO` method), 301  
`optical_flow` (`easycv.core.standard_fields.InputDataFields` attribute), 185  
`OptimizerHook` (class in `easycv.hooks.optimizer_hook`), 137  
`original_image` (`easycv.core.standard_fields.InputDataFields` attribute), 184, 185  
`original_image_shape` (`easycv.core.standard_fields.InputDataFields` attribute), 186  
`original_instance_masks` (`easycv.core.standard_fields.InputDataFields` attribute), 186  
`FileProgress()` (in module `easycv.file.utils`), 304  
`OSSFile` (class in `easycv.file.file_io`), 304  
`OSSSyncHook` (class in `easycv.hooks.oss_sync_hook`), 137  
`FileChannels` (`easycv.models.utils.conv_ws.ConvWS2d` attribute), 277  
`FilePad` (`easycv.models.utils.conv_ws.ConvWS2d` attribute), 277  
`OutputHook` (class in `easycv.predictors.pose_predictor`), 152  
`Params` (class in `easycv.datasets.shared.pipelines.third_transforms_wrapper`), 120  
`padding` (`easycv.models.utils.conv_ws.ConvWS2d` attribute), 277  
`padding_mode` (`easycv.models.utils.conv_ws.ConvWS2d` attribute), 277  
`param_map` (`easycv.models.detection.yolox.yolox.YOLOX` attribute), 250  
`Params` (class in `easycv.core.evaluation.custom_cocotools.cocoeval`), 156  
`params_to_string()` (in module `easycv.utils.flops_counter`), 291  
`parse_arch()` (`easycv.models.backbones.hrnet.HRNet` method), 203  
`parse_list_file()` (`easycv.datasets.classification.data_sources.ClsSource` static method), 43  
`parse_list_file()` (`easycv.datasets.classification.data_sources.image_list` static method), 48  
`parse_list_file()` (`easycv.datasets.selfsup.data_sources.image_list.SSLSourceImage` static method), 110  
`parse_list_file()` (`easycv.datasets.selfsup.data_sources.SSLSourceImage` static method), 109  
`parse_raw()` (in module `easycv.datasets.detection.data_sources.raw`), 73  
`parse_xml()` (in module `easycv.datasets.detection.data_sources.voc`), 74  
`parser_manifest_row_str()` (in module `easycv.datasets.detection.data_sources.pai_format`), 72  
`PatchEmbed` (class in `easycv.models.backbones.swin_transformer_dynamic`), 233  
`PatchEmbed` (class in `easycv.models.backbones.vit_transformer_dynamic`), 237  
`PatchEmbedding` (class in `easycv.models.backbones.shuffle_transformer`), 227  
`patchify()` (`easycv.models.selfsup.mae.MAE` method), 265  
`PatchMerging` (class in `easycv.models.backbones.shuffle_transformer`), 227

|                                                               |                                                                                        |     |
|---------------------------------------------------------------|----------------------------------------------------------------------------------------|-----|
| PatchMerging                                                  | (class in module)                                                                      | 73  |
| easyrcv.models.backbones.swin_transformer_dynamic_postprocess | (class in easyrcv.datasets.classification.pipelines.auto_augment)                      | 232 |
| PILToTensor                                                   | (class in postprocess) (in module)                                                     | 120 |
| easyrcv.datasets.shared.pipelines.third_transforms_wrapper    | (class in easyrcv.models.detection.utils.boxes)                                        | 248 |
| pin_memory                                                    | (easyrcv.datasets.loader.build_loader.InfiniteDataLoader method)                       | 72  |
| attribute                                                     | pre_pipeline() (easyrcv.datasets.detection.data_sources.coco.DetSourceCoco method)     | 98  |
| PlainNet                                                      | (class in easyrcv.models.backbones.genet)                                              | 200 |
| PlainNetBasicBlockClass                                       | (class in predict) (easyrcv.predictors.classifier.TorchClassifier method)              | 193 |
| POOL_DIMS                                                     | (easyrcv.models.utils.multi_pooling.MultiAvgPooling method)                            | 280 |
| attribute                                                     | predict() (easyrcv.predictors.detector.TorchFaceDetector method)                       | 146 |
| POOL_DIMS                                                     | (easyrcv.models.utils.multi_pooling.MultiPooling method)                               | 280 |
| attribute                                                     | predict() (easyrcv.predictors.detector.TorchYoloXClassifierPredictor method)           | 146 |
| pool_flops_counter_hook()                                     | (in module predict) (easyrcv.predictors.detector.TorchYoloXPredictor method)           | 292 |
| easyrcv.utils.flops_counter                                   | predict() (easyrcv.predictors.feature_extractor.TorchFaceAttrExtractor method)         | 150 |
| POOL_PARAMS                                                   | (easyrcv.models.utils.multi_pooling.MultiAvgPooling method)                            | 280 |
| attribute                                                     | predict() (easyrcv.predictors.feature_extractor.TorchFaceFeatureExtractor method)      | 148 |
| POOL_PARAMS                                                   | (easyrcv.models.utils.multi_pooling.MultiPooling method)                               | 279 |
| attribute                                                     | predict() (easyrcv.predictors.feature_extractor.TorchFeatureExtractor method)          | 147 |
| POOL_SIZES                                                    | (easyrcv.models.utils.multi_pooling.MultiAvgPooling method)                            | 280 |
| attribute                                                     | predict() (easyrcv.predictors.feature_extractor.TorchMultiFaceFeatureExtractor method) | 149 |
| POOL_SIZES                                                    | (easyrcv.models.utils.multi_pooling.MultiPooling method)                               | 280 |
| attribute                                                     | predict() (easyrcv.predictors.interface.PredictorInterface method)                     | 150 |
| pose_pck_accuracy()                                           | (in module predict) (easyrcv.predictors.interface.PredictorInterfaceV2 method)         | 169 |
| easyrcv.core.evaluation.top_down_eval                         | predict() (easyrcv.predictors.pose_predictor.TorchPoseTopDownPredictor method)         | 152 |
| PoseCollect                                                   | (class in easyrcv.datasets.pose.pipelines)                                             | 103 |
| predict                                                       | (easyrcv.predictors.pose_predictor.TorchPoseTopDownPredictor method)                   | 153 |
| PoseCollect                                                   | (class in predict_batch) (easyrcv.predictors.base.Predictor method)                    | 106 |
| easyrcv.datasets.pose.pipelines.transforms                    | Predictor (class in easyrcv.predictors.base)                                           | 143 |
| PoseTopDownDataset                                            | (class in easyrcv.datasets.pose)                                                       | 100 |
| predict_batch()                                               | (easyrcv.predictors.interface)                                                         | 150 |
| PoseTopDownDataset                                            | (class in PredictorInterfaceV2) (class in)                                             | 109 |
| easyrcv.datasets.pose.top_down                                | prefetch_factor (easyrcv.datasets.loader.build_loader.InfiniteDataLoader attribute)    | 101 |
| PoseTopDownSource                                             | (class in)                                                                             | 103 |
| easyrcv.datasets.pose.data_sources                            | prepare_train_img()                                                                    | 101 |
| PoseTopDownSource                                             | (class in)                                                                             | 103 |
| easyrcv.datasets.pose.data_sources.top_down                   | (easyrcv.datasets.detection.data_sources.coco.DetSourceCoco method)                    | 103 |
| PoseTopDownSourceCoco                                         | (class in)                                                                             | 100 |
| easyrcv.datasets.pose.data_sources.coco                       | prepare_train_img()                                                                    | 102 |
| PoseTopDownSourceCoco                                         | (class in)                                                                             | 102 |
| easyrcv.datasets.pose.data_sources.coco                       | (easyrcv.datasets.detection.data_sources.DetSourceCoco method)                         | 102 |
| PositionalEncodingFourier                                     | (class in)                                                                             | 239 |
| easyrcv.models.backbones.xcit_transformer                     | preprocess() (easyrcv.predictors.base.Predictor method)                                | 239 |
| post_dark_udp()                                               | (in module PrettyParams) (in module easyrcv.utils.json_utils)                          | 170 |
| easyrcv.core.evaluation.top_down_eval                         | print_model_with_flops() (in module)                                                   | 170 |
| post_process_fn()                                             | (easyrcv.datasets.detection.data_sources.coco.DetSourceCoco method)                    | 70  |
| post_process_fn()                                             | (easyrcv.datasets.detection.data_sources.raw.DetSourceRaw method)                      | 70  |



process\_det\_results() 111  
 (easycv.predictors.pose\_predictor.TorchPoseTopDownPredictor (class in  
 method), 153 easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 process\_polygons() (easycv.datasets.detection.pipelines.LoadAnnotations  
 method), 83 RandomChoice (class in  
 process\_polygons() (easycv.datasets.detection.pipelines.mm\_transforms.LoadAnnotations pipelines.third\_transforms\_wrapper),  
 method), 92 122  
 profile\_time() (in module easycv.utils.profiling), 295 RandomCrop (class in easycv.datasets.shared.pipelines.third\_transforms\_wr  
 proposal\_boxes (easycv.core.standard\_fields.InputDataFields 122  
 attribute), 184, 185 RandomEqualize (class in  
 proposal\_objectness easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 (easycv.core.standard\_fields.InputDataFields 122  
 attribute), 184, 186 RandomErasing (class in  
 pseudo\_dist\_init() (in module easycv.utils.test\_util), easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 297 122  
 put\_text() (in module randomErasing() (in module  
 easycv.core.visualization.image), 182 easycv.utils.preprocess\_function), 295  
 PytorchImageModelWrapper (class in RandomGrayscale (class in  
 easycv.models.backbones.pytorch\_image\_models\_wrapper) easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 215 123  
 random\_gray\_scale() (in module  
 easycv.utils.preprocess\_function), 295  
 RandAugment (class in RandomHorizontalFlip (class in  
 easycv.datasets.shared.pipelines.third\_transforms\_wrapper) easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 120 123  
 random\_masking() (easycv.models.backbones.mae\_vit\_transformer.EncoderAutoencoderViT class  
 method), 210 easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 random\_negative() (in module 123  
 easycv.datasets.classification.pipelines.auto\_augment) RandomOrder (class in  
 54 easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 random\_sample() (easycv.datasets.detection.pipelines.mm\_transforms.MMResize  
 static method), 89 RandomPerspective (class in  
 random\_sample() (easycv.datasets.detection.pipelines.MMResize easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 static method), 79 123  
 random\_sample\_ratio() RandomPosterize (class in  
 (easycv.datasets.detection.pipelines.mm\_transforms.MMResize easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 static method), 89 124  
 random\_sample\_ratio() RandomResizedCrop (class in  
 (easycv.datasets.detection.pipelines.MMResize easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 static method), 80 124  
 random\_select() (easycv.datasets.detection.pipelines.mm\_transforms.MMResize (class in  
 static method), 88 easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 random\_select() (easycv.datasets.detection.pipelines.MMResize 124  
 static method), 79 RandomSolarize (class in  
 RandomAdjustSharpness (class in easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 easycv.datasets.shared.pipelines.third\_transforms\_wrapper) 124  
 121 RandomVerticalFlip (class in  
 RandomAffine (class in easycv.datasets.shared.pipelines.third\_transforms\_wrapper),  
 easycv.datasets.shared.pipelines.third\_transforms\_wrapper) 125  
 121 Ranger (class in easycv.core.optimizer.ranger), 173  
 RandomAppliedTrans (class in RawDataset (class in easycv.datasets.shared), 113  
 easycv.datasets.selfsup.pipelines), 111 RawDataset (class in easycv.datasets.shared.raw), 130  
 RandomAppliedTrans (class in re\_remote (easycv.file.base.IOBase attribute), 299  
 easycv.datasets.selfsup.pipelines.transforms), rebuild\_config() (in module

*easycv.utils.config\_tools*), 290  
**reduction** (*easycv.models.loss.pytorch\_metric\_learning.FocalLoss2d* attribute), 256  
**register()** (*easycv.file.base.IOBase* static method), 299  
**register()** (*easycv.predictors.pose\_predictor.OutputHook* method), 152  
**register\_default\_best\_metric()** (*easycv.core.evaluation.metric\_registry.MetricRegistry* method), 168  
**register\_module()** (*easycv.utils.registry.Registry* method), 296  
**Registry** (class in *easycv.utils.registry*), 296  
**RelativeLocNeck** (class in *easycv.models.selfsup.necks*), 272  
**ReLU** (class in *easycv.models.backbones.genet*), 196  
**relu\_flops\_counter\_hook()** (in module *easycv.utils.flops\_counter*), 292  
**remove()** (*easycv.file.base.IOBase* method), 299  
**remove()** (*easycv.file.base.IOLocal* method), 300  
**remove()** (*easycv.file.file\_io.IO* method), 302  
**remove()** (*easycv.predictors.pose\_predictor.OutputHook* method), 152  
**remove\_batch\_counter\_hook\_function()** (in module *easycv.utils.flops\_counter*), 292  
**remove\_bn\_in\_superblock()** (in module *easycv.models.backbones.genet*), 193  
**remove\_flops\_counter\_hook\_function()** (in module *easycv.utils.flops\_counter*), 292  
**remove\_flops\_mask()** (in module *easycv.utils.flops\_counter*), 291  
**RepeatDataset** (class in *easycv.datasets.shared*), 112  
**RepeatDataset** (class in *easycv.datasets.shared.dataset\_wrappers*), 129  
**replace\_data\_for\_test()** (in module *easycv.utils.test\_util*), 296  
**ResBlock** (class in *easycv.models.backbones.genet*), 197  
**reset\_classifier()** (*easycv.models.backbones.shuffle\_transformer.build\_loader.InfiniteDataLoader* method), 228  
**reset\_flops\_count()** (in module *easycv.utils.flops\_counter*), 291  
**reset\_reader()** (*easycv.datasets.shared.odps\_reader.OdpsReader* method), 130  
**reset\_reader()** (*easycv.datasets.shared.OdpsReader* method), 113  
**Resize** (class in *easycv.datasets.shared.pipelines.third\_transformer.wrapper*), 125  
**resize\_tensor()** (in module *easycv.models.utils.ops*), 282  
**ResLayer** (class in *easycv.models.backbones.network\_blocks*), 213  
**ResLayer** (class in *easycv.models.utils.res\_layer*), 282  
**ResNeSt** (class in *easycv.models.backbones.resnest*), 218  
**ResNet** (class in *easycv.models.backbones.resnet*), 220  
**ResNetJIT** (class in *easycv.models.backbones.resnet\_jit*), 222  
**ResNeXt** (class in *easycv.models.backbones.resnext*), 224  
**results2json()** (*easycv.datasets.detection.DetImagesMixDataset* method), 67  
**results2json()** (*easycv.datasets.detection.mix.DetImagesMixDataset* method), 94  
**resume()** (*easycv.runner.ev\_runner.EVRunner* method), 306  
**RetrivalNeck** (class in *easycv.models.classification.necks*), 245  
**RetrivalTopKEvaluator** (class in *easycv.core.evaluation.retrival\_topk\_eval*), 169  
**rgetattr()** (in module *easycv.predictors.pose\_predictor*), 152  
**rmtree()** (*easycv.file.base.IOBase* method), 299  
**rmtree()** (*easycv.file.base.IOLocal* method), 300  
**rmtree()** (*easycv.file.file\_io.IO* method), 303  
**Rotate** (class in *easycv.datasets.classification.pipelines.auto\_augment*), 60  
**rotate\_point()** (in module *easycv.core.post\_processing*), 175  
**rotate\_point()** (in module *easycv.core.post\_processing.pose\_transforms*), 180  
**rSoftMax** (class in *easycv.models.backbones.resnest*), 216  
**run\_in\_subprocess()** (in module *easycv.utils.test\_util*), 296  
**run\_iter()** (*easycv.runner.ev\_runner.EVRunner* method), 305  
**RunAsSubprocess()** (in module *easycv.utils.test\_util*), 296

## S

**safe\_copy()** (*easycv.file.file\_io.IO* method), 301  
**sampler\_shuffle\_transformer.build\_loader.InfiniteDataLoader** (attribute), 98  
**save\_checkpoint()** (*easycv.runner.ev\_runner.EVRunner* method), 305  
**save\_checkpoint()** (in module *easycv.utils.checkpoint*), 288  
**Scale** (class in *easycv.models.utils.scale*), 283  
**scale\_coords()** (in module *easycv.utils.bbox\_util*), 288  
**seek()** (*easycv.file.file\_io.OSSFile* method), 304  
**Sequential** (class in *easycv.models.backbones.genet*), 197  
**serialize\_tensor()** (in module *easycv.apis.test*), 38  
**set\_data\_loader\_workid()** (in module *easycv.datasets.shared.odps\_reader*), 129  
**set\_data\_loader\_worknum()** (in module *easycv.datasets.shared.odps\_reader*), 129

**set\_epoch()** (*easycv.datasets.loader.DistributedGivenIterationSampler* method), 97  
**set\_epoch()** (*easycv.datasets.loader.DistributedGroupSampler* method), 96  
**set\_epoch()** (*easycv.datasets.loader.sampler.DistributedGroupSampler* method), 100  
**set\_epoch()** (*easycv.datasets.loader.sampler.DistributedGroupSampler* method), 99  
**set\_oss\_env()** (in module *easycv.file.file\_io*), 300  
**set\_random\_seed()** (in module *easycv.apis.train*), 38  
**set\_uniform\_indices()** (*easycv.datasets.loader.DistributedGivenIterationSampler* method), 97  
**set\_uniform\_indices()** (*easycv.datasets.loader.sampler.DistributedGivenIterationSampler* method), 99  
**set\_uniform\_indices()** (*easycv.datasets.loader.sampler.DistributedSampler* method), 99  
**setDetParams()** (*easycv.core.evaluation.custom\_cocotools.source\_detection* method), 156  
**setKpParams()** (*easycv.core.evaluation.custom\_cocotools.source\_detection* method), 156  
**Sharpness** (class in *easycv.datasets.classification.pipelines.source\_detection*), 62  
**Shear** (class in *easycv.datasets.classification.pipelines.auto\_augment*), 58  
**show\_result()** (*easycv.models.base.BaseModel* method), 285  
**show\_result()** (*easycv.models.pose.top\_down.TopDown* method), 262  
**shuffletrans\_base\_p4\_w7\_224()** (in module *easycv.models.backbones.shuffle\_transformer*), 229  
**shuffletrans\_small\_p4\_w7\_224()** (in module *easycv.models.backbones.shuffle\_transformer*), 229  
**shuffletrans\_tiny\_p4\_w7\_224()** (in module *easycv.models.backbones.shuffle\_transformer*), 229  
**ShuffleTransformer** (class in *easycv.models.backbones.shuffle\_transformer*), 228  
**ShuffleUnit** (class in *easycv.models.backbones.lightrnet*), 207  
**SiLU** (class in *easycv.models.backbones.network\_blocks*), 212  
**SimCLR** (class in *easycv.models.selfsup.simclr*), 273  
**single\_cpu\_test()** (in module *easycv.apis.test*), 37  
**single\_gpu\_test()** (in module *easycv.apis.test*), 37  
**size()** (*easycv.file.base.IOBase* method), 299  
**size()** (*easycv.file.base.IOLocal* method), 300  
**size()** (*easycv.file.file\_io.IO* method), 304  
**Sobel** (class in *easycv.models.utils.sobel*), 283  
**SoftSobelNms** (in module *easycv.core.post\_processing*), 176  
**soft\_nms()** (in module *easycv.core.post\_processing.nms*), 177  
**SoftTargetCrossEntropy** (class in *easycv.models.loss.pytorch\_metric\_learning*), 158  
**Solarization** (class in *easycv.datasets.selfsup.pipelines*), 111  
**Solarization** (class in *easycv.datasets.selfsup.pipelines.transforms*), 112  
**Solarize** (class in *easycv.datasets.classification.pipelines.auto\_augment*), 61  
**Solarize60** (in module *easycv.utils.preprocess\_function*), 295  
**SolarizeAdd** (class in *easycv.datasets.classification.pipelines.auto\_augment*), 61  
**source\_detection** (*easycv.core.standard\_fields.DetectionResultFields* attribute), 186  
**source\_detection** (*easycv.core.standard\_fields.InputDataFields* attribute), 184, 185  
**source\_detection** (*easycv.core.standard\_fields.TfExampleFields* attribute), 187, 189  
**SourceConcat** (class in *easycv.datasets.shared.data\_sources*), 114  
**SourceConcat** (class in *easycv.datasets.shared.data\_sources.concat*), 114  
**SpatialWeighting** (class in *easycv.models.backbones.lightrnet*), 204  
**SplAtConv2d** (class in *easycv.models.backbones.resnest*), 216  
**split\_listfile\_byrank()** (in module *easycv.datasets.classification.data\_sources.utils*), 48  
**SPPBottleneck** (class in *easycv.models.backbones.network\_blocks*), 214  
**SSLSourceImageList** (class in *easycv.datasets.selfsup.data\_sources*), 109  
**SSLSourceImageList** (class in *easycv.datasets.selfsup.data\_sources.image\_list*), 110  
**SSLSourceImageNetFeature** (class in *easycv.datasets.selfsup.data\_sources*), 109  
**SSLSourceImageNetFeature** (class in *easycv.datasets.selfsup.data\_sources.imagenet\_feature*), 110  
**StageModule** (class in *easycv.models.backbones.shuffle\_transformer*), 227  
**start\_flops\_count()** (in module

- easycv.utils.flops\_counter*), 291
- Stem* (class in *easycv.models.backbones.lightrnet*), 206
- step()* (*easycv.core.optimizer.lars.LARS* method), 172
- step()* (*easycv.core.optimizer.ranger.Ranger* method), 173
- stop\_flops\_count()* (in module *easycv.utils.flops\_counter*), 291
- stride* (*easycv.models.utils.conv\_ws.ConvWS2d* attribute), 277
- summarize()* (*easycv.core.evaluation.custom\_cocotools.cocoeval.COCOEval* method), 156
- summarize\_per\_category()* (*easycv.core.evaluation.custom\_cocotools.cocoeval.COCOEval* method), 156
- SuperResK1DW* (class in *easycv.models.backbones.genet*), 199
- SuperResK1DWK1* (class in *easycv.models.backbones.genet*), 199
- SuperResK1KX* (class in *easycv.models.backbones.genet*), 198
- SuperResK1KXK1* (class in *easycv.models.backbones.genet*), 198
- SuperResKXXKX* (class in *easycv.models.backbones.genet*), 197
- SUPPORT\_DETECTION\_PREDICTORS* (*easycv.predictors.pose\_predictor.TorchPoseTopDownPredictorWithDetection* attribute), 153
- SWAV* (class in *easycv.models.selfsup.swav*), 274
- SWAVHook* (class in *easycv.hooks.swav\_hook*), 138
- SwinTransformer* (class in *easycv.models.backbones.swin\_transformer\_dynamic*), 234
- SwinTransformerBlock* (class in *easycv.models.backbones.swin\_transformer\_dynamic*), 230
- SyncIBN* (class in *easycv.models.utils.norm*), 280
- SyncNormHook* (class in *easycv.hooks.sync\_norm\_hook*), 139
- SyncRandomSizeHook* (class in *easycv.hooks.sync\_random\_size\_hook*), 139
- ## T
- TenCrop* (class in *easycv.datasets.shared.pipelines.third\_transforms\_wrapper*), 125
- tensor2imgs()* (in module *easycv.utils.misc*), 295
- tensor2obj()* (in module *easycv.utils.dist\_utils*), 290
- TensorboardLoggerHookV2* (class in *easycv.hooks.tensorboard*), 140
- TfExampleFields* (class in *easycv.core.standard\_fields*), 187
- time\_synchronized()* (in module *easycv.utils.profiling*), 295
- TIMEHook* (class in *easycv.hooks.show\_time\_hook*), 138
- timeout* (*easycv.datasets.loader.build\_loader.InfiniteDataLoader* attribute), 98
- tmpdir* (*easycv.hooks.eval\_hook.DistEvalHook* attribute), 136
- to\_tensor()* (in module *easycv.datasets.shared.pipelines.format*), 116
- TopDown* (class in *easycv.models.pose.top\_down*), 261
- TopDownAffine* (class in *easycv.datasets.pose.pipelines*), 104
- TopDownAffine* (class in *easycv.datasets.pose.pipelines.transforms*), 107
- TopDownGenerateTarget* (class in *easycv.datasets.pose.pipelines*), 105
- TopDownGenerateTarget* (class in *easycv.datasets.pose.pipelines.transforms*), 107
- TopDownGenerateTargetRegression* (class in *easycv.datasets.pose.pipelines*), 105
- TopDownGenerateTargetRegression* (class in *easycv.datasets.pose.pipelines.transforms*), 108
- TopDownGetRandomScaleRotation* (class in *easycv.datasets.pose.pipelines*), 104
- TopDownGetRandomScaleRotation* (class in *easycv.datasets.pose.pipelines.transforms*), 107
- TopDownHalfBodyTransform* (class in *easycv.datasets.pose.pipelines*), 104
- TopDownHalfBodyTransform* (class in *easycv.datasets.pose.pipelines.transforms*), 106
- TopdownHeatmapBaseHead* (class in *easycv.models.pose.heads.topdown\_heatmap\_base\_head*), 259
- TopdownHeatmapSimpleHead* (class in *easycv.models.pose.heads.topdown\_heatmap\_simple\_head*), 259
- TopDownRandomFlip* (class in *easycv.datasets.pose.pipelines*), 104
- TopDownRandomFlip* (class in *easycv.datasets.pose.pipelines.transforms*), 106
- TopDownRandomTranslation* (class in *easycv.datasets.pose.pipelines*), 105
- TopDownRandomTranslation* (class in *easycv.datasets.pose.pipelines.transforms*), 108
- ToPILImage* (class in *easycv.datasets.shared.pipelines.third\_transforms\_wrapper*), 125
- TorchClassifier* (class in *easycv.predictors.classifier*), 144
- TorchFaceAttrExtractor* (class in *easycv.predictors.feature\_extractor*), 149





- attribute), 195
- training (easycv.models.backbones.genet.MaxPool attribute), 196
- training (easycv.models.backbones.genet.MultiSumBlock attribute), 196
- training (easycv.models.backbones.genet.PlainNet attribute), 200
- training (easycv.models.backbones.genet.PlainNetBasicBlock attribute), 193
- training (easycv.models.backbones.genet.RELU attribute), 197
- training (easycv.models.backbones.genet.ResBlock attribute), 197
- training (easycv.models.backbones.genet.Sequential attribute), 197
- training (easycv.models.backbones.genet.SuperResK1DW attribute), 199
- training (easycv.models.backbones.genet.SuperResK1DWK attribute), 199
- training (easycv.models.backbones.genet.SuperResK1KX attribute), 198
- training (easycv.models.backbones.genet.SuperResK1KXK attribute), 199
- training (easycv.models.backbones.genet.SuperResK1KXKX attribute), 198
- training (easycv.models.backbones.genet.SuperResK1KXKXK attribute), 198
- training (easycv.models.backbones.hrnet.Bottleneck attribute), 201
- training (easycv.models.backbones.hrnet.HRModule attribute), 201
- training (easycv.models.backbones.hrnet.HRNet attribute), 203
- training (easycv.models.backbones.inceptionv3.Inception3 attribute), 204
- training (easycv.models.backbones.lighthrnet.ConditionalConv attribute), 206
- training (easycv.models.backbones.lighthrnet.CrossResolution attribute), 205
- training (easycv.models.backbones.lighthrnet.IterativeHeatmap attribute), 207
- training (easycv.models.backbones.lighthrnet.LiteHRModule attribute), 208
- training (easycv.models.backbones.lighthrnet.LiteHRNet attribute), 210
- training (easycv.models.backbones.lighthrnet.ShuffleUnit attribute), 208
- training (easycv.models.backbones.lighthrnet.SpatialWeighting attribute), 205
- training (easycv.models.backbones.lighthrnet.Stem attribute), 207
- training (easycv.models.backbones.mae\_vit\_transformer.MultiHeadCrossAttention attribute), 211
- training (easycv.models.backbones.mnasnet.MNASNet attribute), 211
- training (easycv.models.backbones.mobilenetv2.MobileNetV2 attribute), 212
- training (easycv.models.backbones.network\_blocks.BaseConv attribute), 213
- training (easycv.models.backbones.network\_blocks.Bottleneck attribute), 213
- training (easycv.models.backbones.network\_blocks.CSPLayer attribute), 215
- training (easycv.models.backbones.network\_blocks.DWConv attribute), 213
- training (easycv.models.backbones.network\_blocks.Focus attribute), 215
- training (easycv.models.backbones.network\_blocks.HSiLU attribute), 212
- training (easycv.models.backbones.network\_blocks.ResLayer attribute), 214
- training (easycv.models.backbones.network\_blocks.SiLU attribute), 212
- training (easycv.models.backbones.network\_blocks.SPPBottleneck attribute), 214
- training (easycv.models.backbones.pytorch\_image\_models\_wrapper.PytorchImageModelsWrapper attribute), 216
- training (easycv.models.backbones.resnest.Bottleneck attribute), 217
- training (easycv.models.backbones.resnest.GlobalAvgPool2d attribute), 217
- training (easycv.models.backbones.resnest.ResNeSt attribute), 218
- training (easycv.models.backbones.resnest.rSoftMax attribute), 217
- training (easycv.models.backbones.resnest.SplAtConv2d attribute), 216
- training (easycv.models.backbones.resnet.BasicBlock attribute), 219
- training (easycv.models.backbones.resnet.Bottleneck attribute), 219
- training (easycv.models.backbones.resnet.ResNet attribute), 221
- training (easycv.models.backbones.resnet\_jit.BasicBlock attribute), 222
- training (easycv.models.backbones.resnet\_jit.Bottleneck attribute), 222
- training (easycv.models.backbones.resnet\_jit.ResNetJIT attribute), 223
- training (easycv.models.backbones.resnext.Bottleneck attribute), 224
- training (easycv.models.backbones.resnext.ResNeXt attribute), 225
- training (easycv.models.backbones.shuffle\_transformer.Attention attribute), 226
- training (easycv.models.backbones.shuffle\_transformer.Block attribute), 226
- training (easycv.models.backbones.shuffle\_transformer.Mlp attribute), 226
- training (easycv.models.backbones.shuffle\_transformer.PatchEmbedding attribute), 226

attribute), 228  
 training (easycv.models.backbones.shuffle\_transformer.PatchEmbed attribute), 227  
 training (easycv.models.backbones.shuffle\_transformer.ShuffleTransformer attribute), 229  
 training (easycv.models.backbones.shuffle\_transformer.Stage attribute), 227  
 training (easycv.models.backbones.swin\_transformer\_dynamic attribute), 233  
 training (easycv.models.backbones.swin\_transformer\_dynamic attribute), 229  
 training (easycv.models.backbones.swin\_transformer\_dynamic attribute), 234  
 training (easycv.models.backbones.swin\_transformer\_dynamic attribute), 232  
 training (easycv.models.backbones.swin\_transformer\_dynamic attribute), 235  
 training (easycv.models.backbones.swin\_transformer\_dynamic attribute), 232  
 training (easycv.models.backbones.swin\_transformer\_dynamic attribute), 230  
 training (easycv.models.backbones.vit\_transformer\_dynamic attribute), 237  
 training (easycv.models.backbones.vit\_transformer\_dynamic attribute), 237  
 training (easycv.models.backbones.vit\_transformer\_dynamic attribute), 236  
 training (easycv.models.backbones.vit\_transformer\_dynamic attribute), 236  
 training (easycv.models.backbones.vit\_transformer\_dynamic attribute), 237  
 training (easycv.models.backbones.vit\_transformer\_dynamic attribute), 238  
 training (easycv.models.backbones.xcit\_transformer.Classifier attribute), 240  
 training (easycv.models.backbones.xcit\_transformer.Classifier attribute), 241  
 training (easycv.models.backbones.xcit\_transformer.Conv attribute), 240  
 training (easycv.models.backbones.xcit\_transformer.LPI attribute), 240  
 training (easycv.models.backbones.xcit\_transformer.Position attribute), 239  
 training (easycv.models.backbones.xcit\_transformer.XCA attribute), 241  
 training (easycv.models.backbones.xcit\_transformer.XCAB attribute), 242  
 training (easycv.models.backbones.xcit\_transformer.XCiT attribute), 243  
 training (easycv.models.base.BaseModel attribute), 285  
 training (easycv.models.classification.classification.Classifier attribute), 244  
 training (easycv.models.classification.necks.FaceIDNeck attribute), 246  
 training (easycv.models.classification.necks.HRFuseScales attribute), 247  
 training (easycv.models.classification.necks.LinearNeck attribute), 245  
 training (easycv.models.classification.necks.MultiLinearNeck attribute), 247  
 training (easycv.models.classification.necks.Re retrievalNeck attribute), 246  
 training (easycv.models.detection.yolox.yolo\_head.YOLOXHead attribute), 249  
 training (easycv.models.detection.yolox.yolo\_pafpn.YOLOPAFPN attribute), 249  
 training (easycv.models.detection.yolox.yolox.YOLOX attribute), 250  
 training (easycv.models.detection.yolox\_edge.yolox\_edge.YOLOX\_EDGE attribute), 251  
 training (easycv.models.detection.yolox\_head.cls\_head.ClsHead attribute), 252  
 training (easycv.models.detection.yolox\_head.contrastive\_head.ContrastiveHead attribute), 252  
 training (easycv.models.detection.yolox\_head.contrastive\_head.DebiasContrastiveHead attribute), 252  
 training (easycv.models.detection.yolox\_head.latent\_pred\_head.LatentClsHead attribute), 253  
 training (easycv.models.detection.yolox\_head.latent\_pred\_head.LatentPredictHead attribute), 253  
 training (easycv.models.detection.yolox\_head.mp\_metric\_head.MpMetricHead attribute), 254  
 training (easycv.models.detection.yolox\_head.multi\_cls\_head.MultiClsHead attribute), 255  
 training (easycv.models.loss.iou\_loss.IOULoss attribute), 255  
 training (easycv.models.loss.mse\_loss.JointsMSELoss attribute), 255  
 training (easycv.models.loss.pytorch\_metric\_learning.AMSoftmaxLoss attribute), 257  
 training (easycv.models.loss.pytorch\_metric\_learning.CrossEntropyLoss attribute), 257  
 training (easycv.models.loss.pytorch\_metric\_learning.DistributeMSELoss attribute), 256  
 training (easycv.models.loss.pytorch\_metric\_learning.ModelParallelAMS attribute), 258  
 training (easycv.models.loss.pytorch\_metric\_learning.ModelParallelSoft attribute), 258  
 training (easycv.models.loss.pytorch\_metric\_learning.SoftTargetCrossEn attribute), 258  
 training (easycv.models.pose.heads.topdown\_heatmap\_base\_head.Topdown attribute), 259  
 training (easycv.models.pose.heads.topdown\_heatmap\_simple\_head.Topdown attribute), 261  
 training (easycv.models.pose.top\_down.TopDown attribute), 262  
 training (easycv.models.selfsup.byol.BYOL attribute),

- 263
- training (*easycv.models.selfsup.dino.DINO* attribute), 265
- training (*easycv.models.selfsup.dino.DINOLoss* attribute), 264
- training (*easycv.models.selfsup.dino.MultiCropWrapper* attribute), 264
- training (*easycv.models.selfsup.mae.MAE* attribute), 266
- training (*easycv.models.selfsup.mixco.MIXCO* attribute), 266
- training (*easycv.models.selfsup.moby.MoBY* attribute), 268
- training (*easycv.models.selfsup.moco.MOCO* attribute), 269
- training (*easycv.models.selfsup.necks.DINONeck* attribute), 269
- training (*easycv.models.selfsup.necks.MAENeck* attribute), 272
- training (*easycv.models.selfsup.necks.MoBYMLP* attribute), 269
- training (*easycv.models.selfsup.necks.NonLinearNeckSimCLR* attribute), 272
- training (*easycv.models.selfsup.necks.NonLinearNeckSwav* attribute), 270
- training (*easycv.models.selfsup.necks.NonLinearNeckV0* attribute), 270
- training (*easycv.models.selfsup.necks.NonLinearNeckV1* attribute), 270
- training (*easycv.models.selfsup.necks.NonLinearNeckV2* attribute), 271
- training (*easycv.models.selfsup.necks.RelativeLocNeck* attribute), 272
- training (*easycv.models.selfsup.simclr.SimCLR* attribute), 273
- training (*easycv.models.selfsup.swav.MultiPrototypes* attribute), 275
- training (*easycv.models.selfsup.swav.SWAV* attribute), 274
- training (*easycv.models.utils.accuracy.Accuracy* attribute), 275
- training (*easycv.models.utils.activation.FReLU* attribute), 276
- training (*easycv.models.utils.conv\_module.ConvModule* attribute), 277
- training (*easycv.models.utils.dist\_utils.DistributedLossWrapper* attribute), 278
- training (*easycv.models.utils.dist\_utils.DistributedMinerWrapper* attribute), 278
- training (*easycv.models.utils.multi\_pooling.GeMPooling* attribute), 279
- training (*easycv.models.utils.multi\_pooling.MultiAvgPooling* attribute), 280
- training (*easycv.models.utils.multi\_pooling.MultiPooling* attribute), 280
- training (*easycv.models.utils.norm.IBN* attribute), 281
- training (*easycv.models.utils.norm.SyncIBN* attribute), 281
- training (*easycv.models.utils.scale.Scale* attribute), 283
- training (*easycv.models.utils.sobel.Sobel* attribute), 283
- transform\_preds() (in module *easycv.core.post\_processing*), 175
- transform\_preds() (in module *easycv.core.post\_processing.pose\_transforms*), 179
- Translate (class in *easycv.datasets.classification.pipelines.auto\_augment*), 59
- transposed (*easycv.models.utils.conv\_ws.ConvWS2d* attribute), 277
- traverse\_replace() (in module *easycv.utils.config\_tools*), 289
- TrivialAugmentWide (class in *easycv.datasets.shared.pipelines.third\_transforms\_wrapper*), 125
- True\_image\_shape (*easycv.core.standard\_fields.InputDataFields* attribute), 186
- true\_image\_shapes (*easycv.core.standard\_fields.InputDataFields* attribute), 185
- trunc\_normal\_() (in module *easycv.models.utils.init\_weights*), 279
- ## U
- unmap() (in module *easycv.utils.misc*), 295
- update() (*easycv.hooks.ema\_hook.ModelEMA* method), 134
- update\_attr() (*easycv.hooks.ema\_hook.ModelEMA* method), 134
- update\_center() (*easycv.models.selfsup.dino.DINOLoss* method), 264
- update\_dynamic\_scale() (*easycv.datasets.detection.DetImagesMixDataset* method), 67
- update\_dynamic\_scale() (*easycv.datasets.detection.mix.DetImagesMixDataset* method), 94
- update\_extract\_list() (*easycv.models.classification.classification.Classification* method), 244
- update\_skip\_type\_keys() (*easycv.datasets.detection.DetImagesMixDataset* method), 67
- update\_skip\_type\_keys() (*easycv.datasets.detection.mix.DetImagesMixDataset* method), 94
- upload\_file() (*easycv.hooks.oss\_sync\_hook.OSSSyncHook* method), 138



- `upsample_flops_counter_hook()` (in module `easycv.utils.flops_counter`), 292
- ## V
- `val()` (`easycv.runner.ev_runner.EVRunner` method), 305
- `val_step()` (`easycv.models.base.BaseModel` method), 285
- `validate_export_config()` (in module `easycv.utils.config_tools`), 290
- `version` (`easycv.predictors.interface.PredictorInterface` attribute), 150
- `version` (`easycv.predictors.interface.PredictorInterfaceV2` attribute), 151
- `vis_pose_result()` (in module `easycv.predictors.pose_predictor`), 153
- `VisionTransformer` (class in `easycv.models.backbones.vit_transformer_dynamic`), 238
- `visualization_log()` (`easycv.hooks.tensorboard.TensorboardLoggerHookV2` method), 140
- `visualization_log()` (`easycv.hooks.wandb.WandbLoggerHookV2` method), 140
- `visualize()` (`easycv.datasets.classification.ClsDataset` method), 41
- `visualize()` (`easycv.datasets.classification.raw.ClsDataset` method), 65
- `visualize()` (`easycv.datasets.detection.DetDataset` method), 66
- `visualize()` (`easycv.datasets.detection.raw.DetDataset` method), 95
- `visualize()` (`easycv.datasets.shared.base.BaseDataset` method), 126
- `visualize()` (`easycv.datasets.shared.BaseDataset` method), 113
- ## W
- `WandbLoggerHookV2` (class in `easycv.hooks.wandb`), 140
- `warp_affine_joints()` (in module `easycv.core.post_processing`), 176
- `warp_affine_joints()` (in module `easycv.core.post_processing.pose_transforms`), 180
- `weight` (`easycv.models.utils.conv_ws.ConvWS2d` attribute), 277
- `width` (`easycv.core.standard_fields.InputDataFields` attribute), 185
- `width` (`easycv.core.standard_fields.TfExampleFields` attribute), 187, 188
- `window_partition()` (in module `easycv.models.backbones.swin_transformer_dynamic`), 229
- `window_reverse()` (in module `easycv.models.backbones.swin_transformer_dynamic`), 229
- `WindowAttention` (class in `easycv.models.backbones.swin_transformer_dynamic`), 230
- `with_keypoint` (`easycv.models.pose.top_down.TopDown` property), 261
- `with_neck` (`easycv.models.pose.top_down.TopDown` property), 261
- `worker_init_fn()` (in module `easycv.datasets.loader.build_loader`), 98
- `wrap_torchvision_transforms()` (in module `easycv.datasets.shared.pipelines.third_transforms_wrapper`), 118
- `write()` (`easycv.file.file_io.OSSFile` method), 304
- ## X
- `XCA` (class in `easycv.models.backbones.xcit_transformer`), 241
- `XCABlock` (class in `easycv.models.backbones.xcit_transformer`), 241
- `XCiT` (class in `easycv.models.backbones.xcit_transformer`), 242
- `xcit_large_24_p8()` (in module `easycv.models.backbones.xcit_transformer`), 243
- `xcit_medium_24_p16()` (in module `easycv.models.backbones.xcit_transformer`), 243
- `xcit_medium_24_p8()` (in module `easycv.models.backbones.xcit_transformer`), 243
- `xcit_small_12_p16()` (in module `easycv.models.backbones.xcit_transformer`), 243
- `xcit_small_12_p8()` (in module `easycv.models.backbones.xcit_transformer`), 243
- `xcit_small_24_p16()` (in module `easycv.models.backbones.xcit_transformer`), 243
- `xcit_small_24_p8()` (in module `easycv.models.backbones.xcit_transformer`), 243
- `xywh2xyxy()` (in module `easycv.utils.bbox_util`), 287
- `xywh2xyxy_coco()` (in module `easycv.utils.bbox_util`), 287
- `xyxy2xywh()` (`easycv.datasets.detection.data_sources.coco.DetSourceCoco` method), 72
- `xyxy2xywh()` (`easycv.datasets.detection.data_sources.DetSourceCoco` method), 68
- `xyxy2xywh()` (in module `easycv.utils.bbox_util`), 287

`xyxy2xywh_coco()` (*in module `easycv.utils.bbox_util`*),  
287  
`xyxy2xywh_with_shape()` (*in module  
`easycv.utils.bbox_util`*), 287

## Y

`YOLOPAFPN` (*class in `easycv.models.detection.yolox.yolo_pafpn`*),  
249  
`YOLOX` (*class in `easycv.models.detection.yolox.yolox`*), 250  
`YOLOX_EDGE` (*class in `easycv.models.detection.yolox_edge.yolox_edge`*),  
251  
`YOLOXHead` (*class in `easycv.models.detection.yolox.yolo_head`*),  
248  
`YOLOXlrUpdaterHook` (*class in  
`easycv.hooks.yolox_lr_hook`*), 140  
`YOLOXModeSwitchHook` (*class in  
`easycv.hooks.yolox_mode_switch_hook`*),  
141