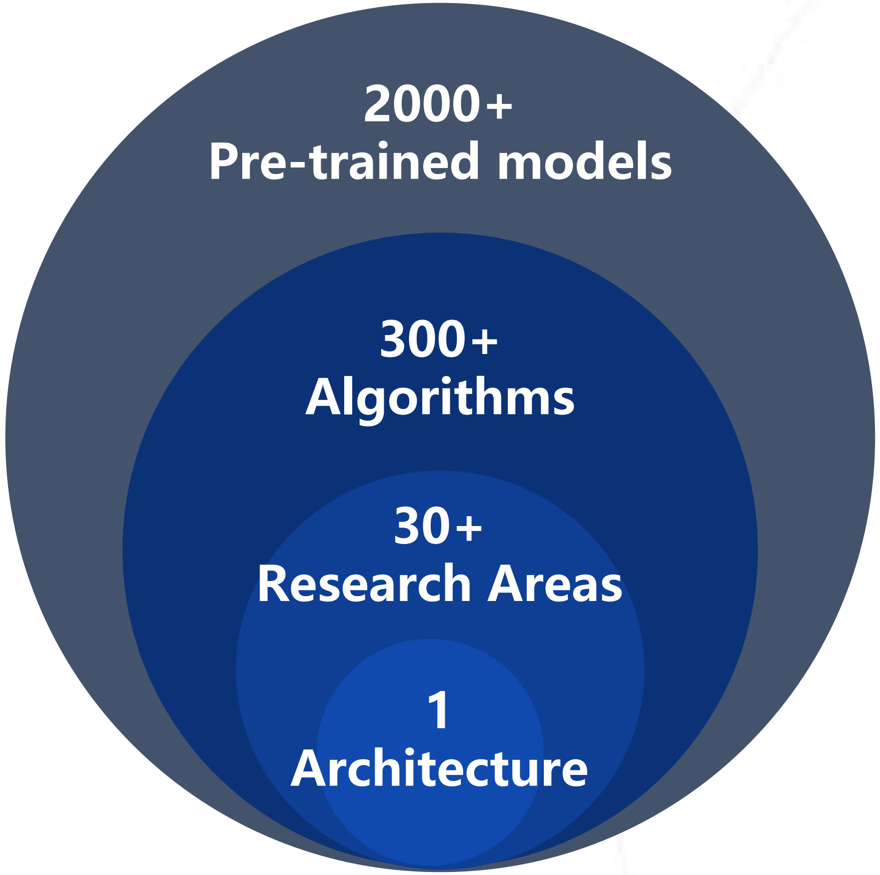
Overview and Recent Updates of OpenMMLab

Kai Chen
Shanghai AI Lab

Contents

Overview



1 Architecture

- A unified architecture for all codebases

30+ Research Areas

- Cover various areas of computer vision

300+ Algorithms

- Implement both classical and most recent algorithms

2000+ Pretrained Models

- Unified benchmark and out-of-box usage

Supported Areas

OpenMMLab

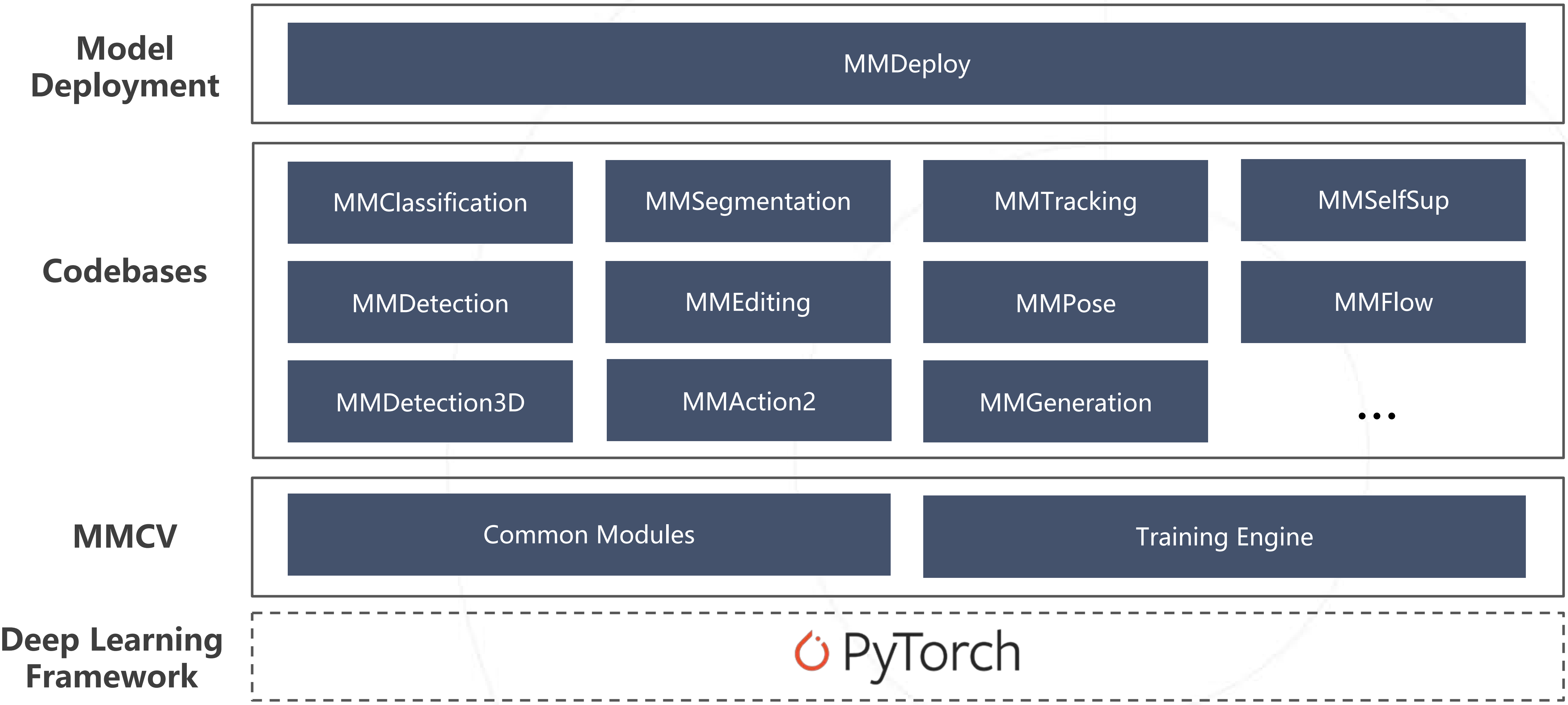
Image Matting
Video Inpainting
Super Resolution
Image Inpainting
Diffusion Models
3D Semantic Segmentation
Self-Supervised Learning
Video Instance Segmentation
Optical Flow Estimation
Key Information Extraction
Knowledge Distillation
Generative Adversarial Network
Video Object Detection
3D Human Parametric Model
Spatio-Temporal Action Detecti
Multi-Object Tracking
Gesture Recognition
Action Localization
Frame Interpolation
3D Object Detection
Pose Estimation
Few-Shot Classification
Panoptic Segmentation
Visual Object Tracking
Semantic Segmentation
Instance Segmentation
Neural Architecture Search
Rotated Object Detection
Object Detection
Few-Shot Detection
Image Classification
Network Pruning
OCR
Image Translation

History

OpenMMLab

- **2018**: MMCV, MMDetection
- **2019**: MMAction, MMSR, MMSkeleton, MMFashion
- **2020-H1**: OpenPCDet, OpenUnReID, OpenSelfSup
- **2020-H2**: MMSegmentation, MMDetection3D, MMClassification, MMAction2, MMEditing, MMPose
- **2021-H1**: MMTracking, MMGeneration, MMOCR
- **2021-H2**: MMFewShot, MMFlow, MMSelfSup, MMHuman3D, MMRazor, MMDeploy
- **2022-H1**: MMRotate
- **Future**: More is coming

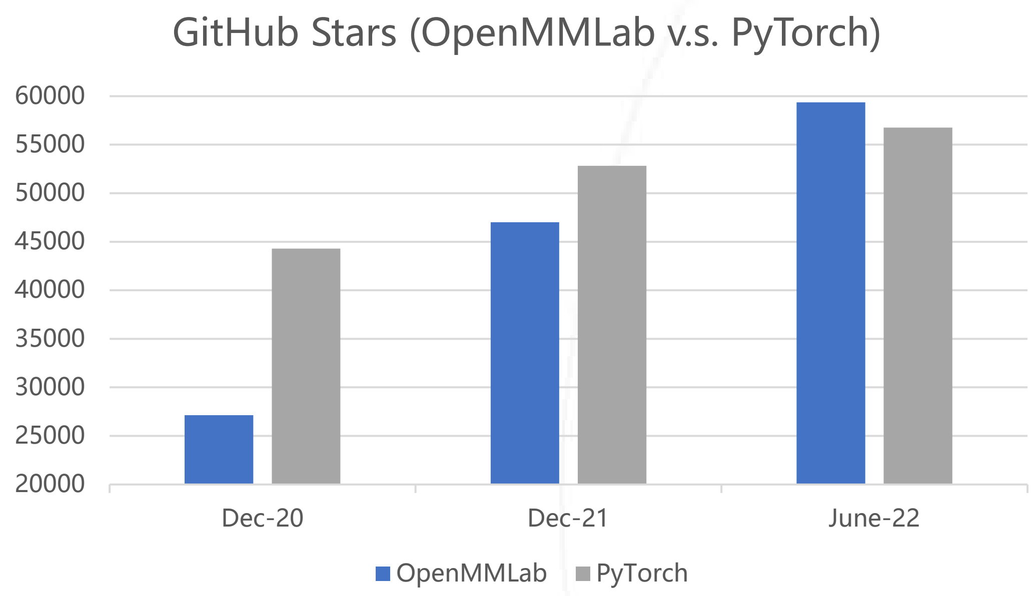
Framework


Model Deployment
MMDeploy
Codebases
MMClassification
MMSegmentation
MMTracking
MMSelfSup
MMDetection
MMEditing
MMPose
MMFlow
MMDetection3D
MMAction2
MMGeneration
...
MMCV
Common Modules
Training Engine
Deep Learning Framework
PyTorch

Community Impact

OpenMMLab

GitHub Stars ~60,000



Users

- 110 countries/regions
- 600 colleges and research institutes
- 1M checkpoint downloads per year

Developers

- 1000+ contributors
- 500k lines of " import mmcv/mmdet/mmseg/…" on GitHub

Academic

- mentioned in 1000+ papers since released
- adopted by 76 papers in CVPR 2022
- 20+ challenge winners

Community Impact

OpenMMLab

Carnegie Mellon University
salesforce
ILLINOIS
清華大學 Tsinghua University
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
浙江大學 ZHEJIANG UNIVERSITY
哈爾濱工業大學 HARBIN INSTITUTE OF TECHNOLOGY
UNIVERSITY of VIRGINIA
TU/e Technische Universiteit Eindhoven University of Technology
ASU Arizona State University
Tencent 腾讯
intel.
Microsoft
ByteDance 字节跳动
Google
NUS National University of Singapore
THE UNIVERSITY of ADELAIDE
ARGO AI
Alibaba Group 阿里巴巴集团
北京大學 PEKING UNIVERSITY
JOHNS HOPKINS UNIVERSITY
UNIVERSITY OF OXFORD
北京航空航天大學 BEIHANG UNIVERSITY
HUAWEI
上海交通大學 SHANGHAI JIAO TONG UNIVERSITY
同濟大學 TONGJI UNIVERSITY
中国科学院大学 University of Chinese Academy of Sciences
THE UNIVERSITY OF SYDNEY
U LISBOA UNIVERSIDADE DE LISBOA
商汤 sensetime

Updates in the Past Year

OpenMMLab

6 new codebases in different areas

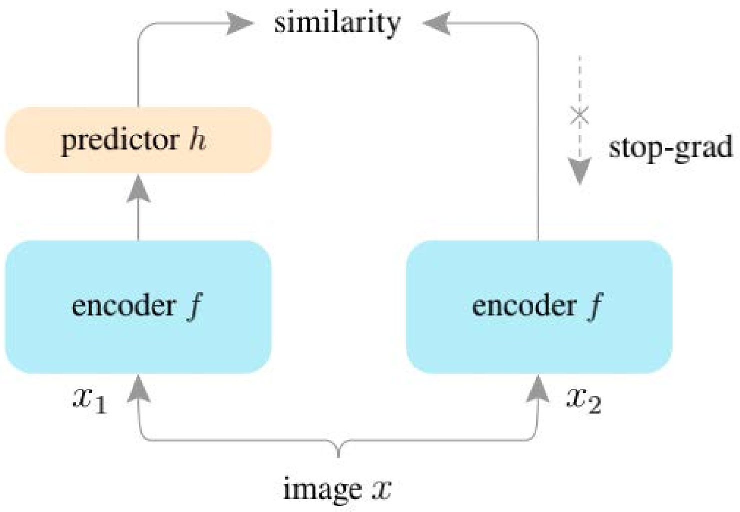
MMSelfSup
MMFewShot
MMFlow
MMRotate
MMRazor
MMHuman3D

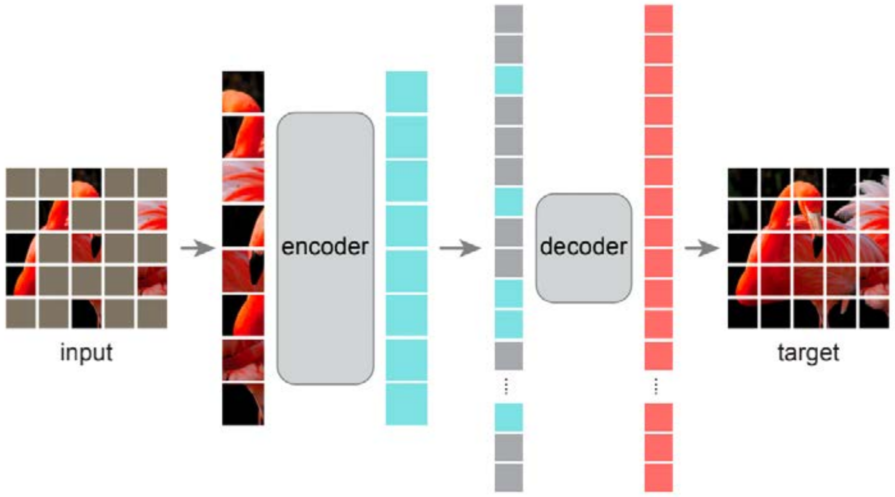
From research to production

MMDeploy

MMSelfSup

OpenMMLab





- ✓ Relative Location
- ✓ Rotation Prediction
- ✓ DeepCluster
- ✓ NPID
- ✓ ODC
- ✓ MoCo v1
- ✓ MoCo v2
- ✓ MoCo v3
- ✓ SimCLR
- ✓ BYOL
- ✓ SwAV
- ✓ DenseCL
- ✓ SimSiam
- ✓ Barlow Twins
- ✓ MAE
- ✓ SimMIM
- ✓ CAE

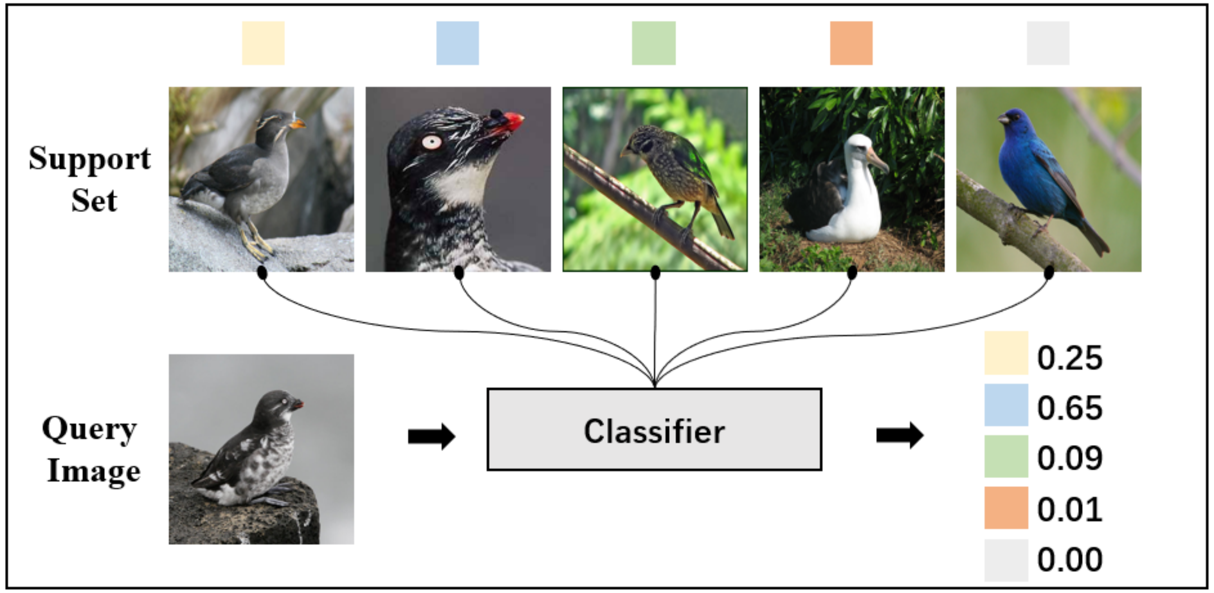
Popular self-supervised learning paradigms and algorithms

Benchmarks and downstream-tasks for evaluation

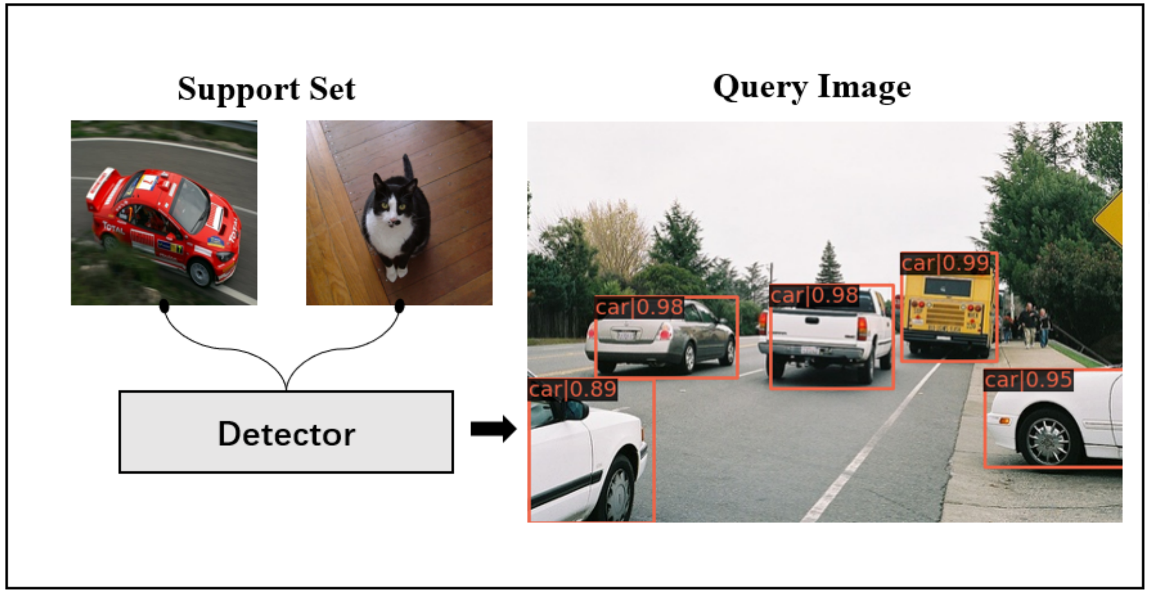
MMFewshot

OpenMMLab

Few Shot Classification



Few Shot Detection



The first codebase that provides unified implementation and evaluation of few shot classification and detection.

Classification	Detection
Baseline Baseline++ NegMargin MatchingNet ProtoNet RelationNet MetaBaseline MAML	TFA FSCE AttentionRPN MetaRCNN FSDetView MPSR

MMFlow

The first systematic toolbox for optical flow estimation.

Supported methods

- ✓ FlowNet
- ✓ FlowNet2
- ✓ PWC-Net
- ✓ LiteFlowNet
- ✓ LiteFlowNet2
- ✓ IRR
- ✓ MaskFlownet
- ✓ RAFT
- ✓ GMA

MMRotate

OpenMMLab

The most powerful and complete toolbox for rotated object detection

- ✓ Rotated RetinaNet-OBB/HBB
- ✓ Rotated FasterRCNN-OBB
- ✓ Rotated RepPoints-OBB
- ✓ Rotated FCOS
- ✓ RoI Transformer
- ✓ Gliding Vertex
- ✓ Rotated ATSS-OBB
- ✓ CSL
- ✓ R3Det
- ✓ S2A-Net
- ✓ ReDet
- ✓ Beyond Bounding-Box
- ✓ Oriented R-CNN
- ✓ GWD
- ✓ KLD
- ✓ SASM
- ✓ KFIoU

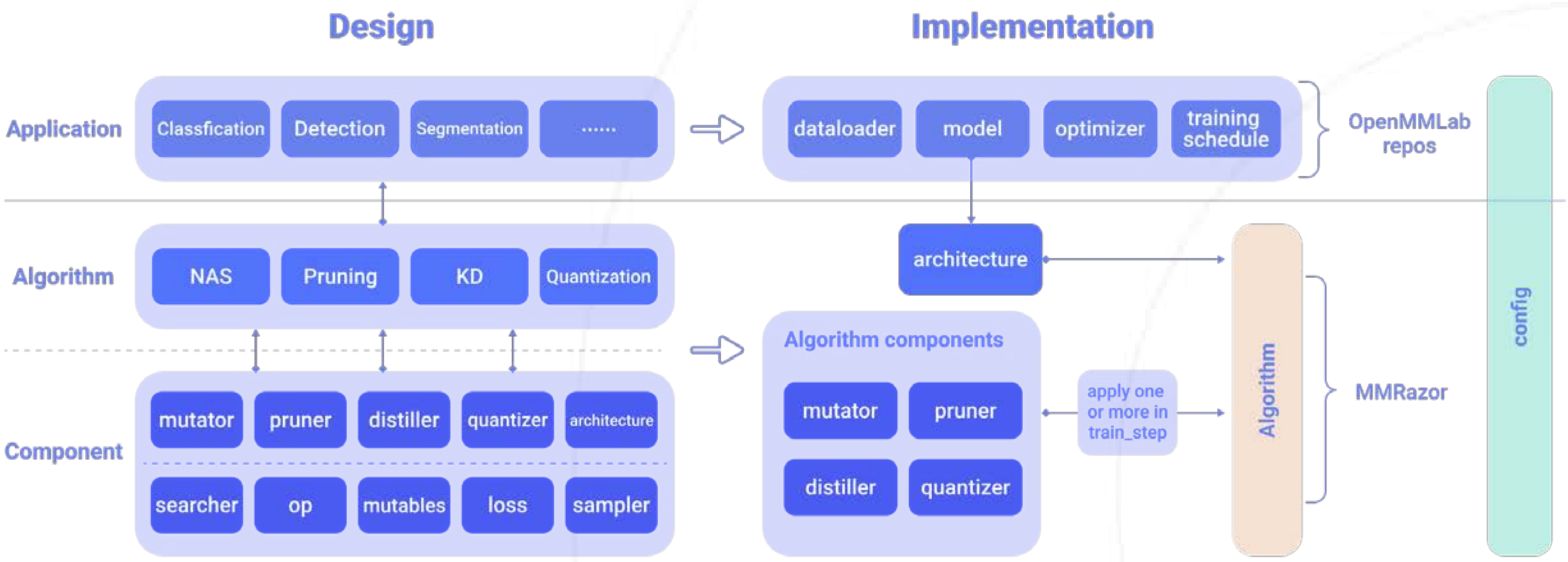
MMhuman3d

Reproducing popular methods with a modular framework

Supporting various datasets with a unified data convention

Versatile visualization toolbox

MMRazor



- ✓ NAS
- ✓ Network Pruning
- ✓ Knowledge Distillation

From Research to Production

OpenMMLab

Training Framework

PyTorch

Intermediate Representation

ONNX

TorchScript

Inference Engine

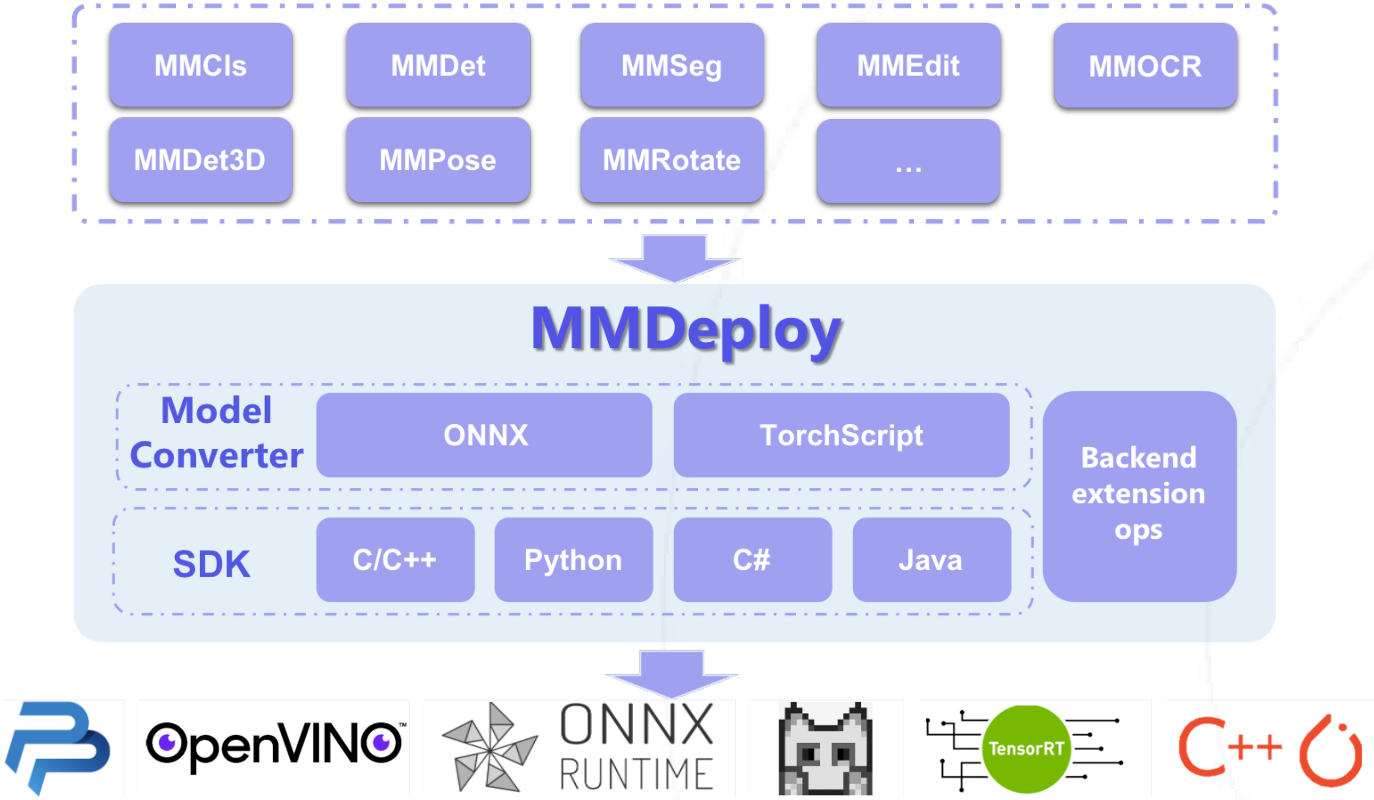
TensorRT

ONNX RUNTIME

OpenVINO

...

MMDeploy



Various inference engines

TensorRT、ONNXRuntime、OpenVINO、ncnn、libtorch、PPL.NN

Multiple platforms

Linux、Windows、Android、macOS（WIP)

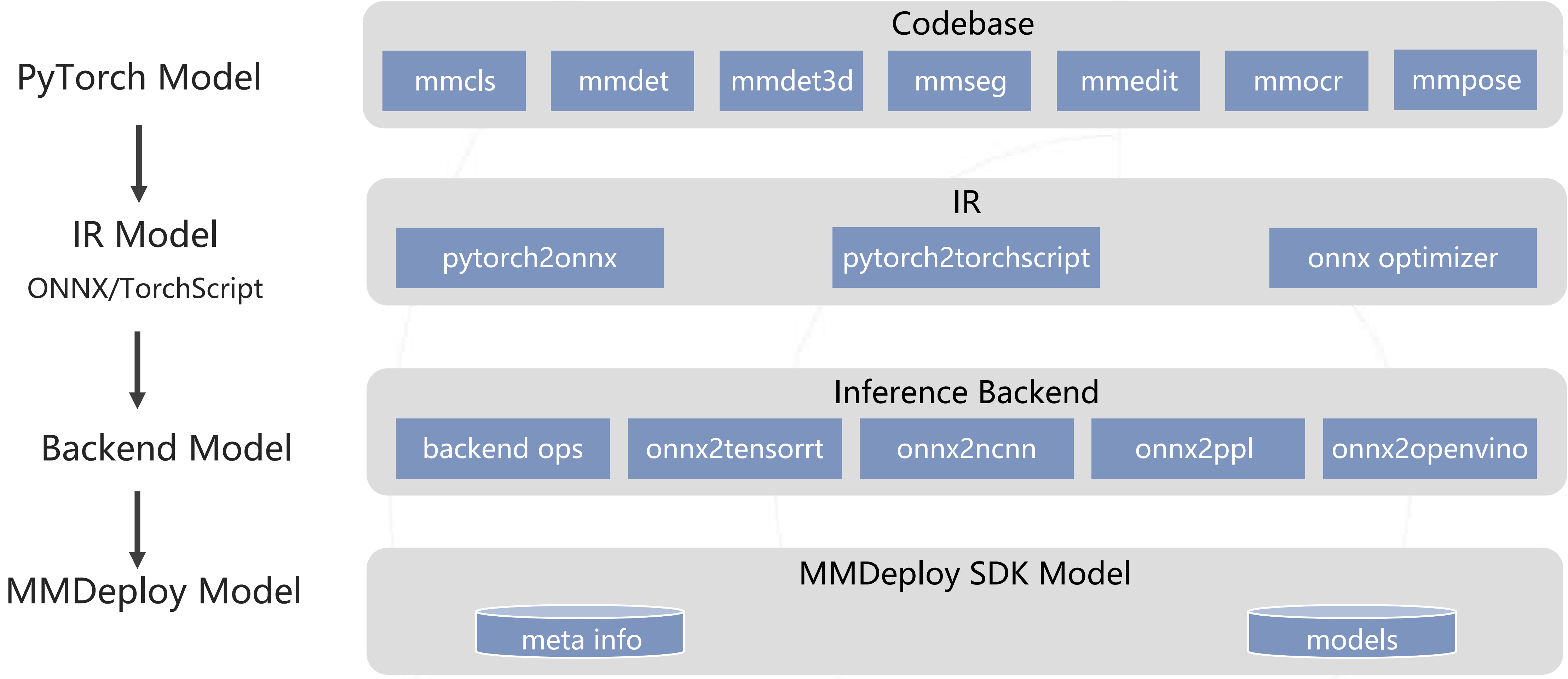
Multiple language support

C/C++、Python、Java、C#

Flexible integration to user system

IR models、Inference engine models、MMDeploy SDK

MMDeploy


PyTorch Model
IR Model
ONNX/TorchScript
Backend Model
MMDeploy Model
Codebase
mmcls
mmdet
mmdet3d
mmseg
mmedit
mmocr
mmpose
IR
pytorch2onnx
pytorch2torchscript
onnx optimizer
Inference Backend
backend ops
onnx2tensorrt
onnx2ncnn
onnx2ppl
onnx2openvino
MMDeploy SDK Model
meta info
models

Usage Example

```
python tools/deploy.py \
  mmdeploy/configs/mmdet/detection/detection_tensorrt_dynamic-320x320-1344x1344.py \
  mmdetection/configs/retinanet/retinanet_r50_fpn_1x_coco.py \
  retinanet_r50_fpn_1x_coco_20200130-c2398f9e.pth \
  mmdetection/demo/demo.jpg \
  --work-dir retinanet/tensorrt \
  --device cuda:0 \
  --dump-info
```

RetinaNet

Model Converter → MMDeploy Model → MMDeploy SDK → NVIDIA

```
retinanet/tensorrt
├── deploy.json
├── detail.json
├── end2end.engine
├── end2end.onnx
├── output_pytorch.jpg
├── output_tensorrt.jpg
└── pipeline.json
```

```python
from mmdeploy_python import Detector
import cv2

model_path='retinanet/tensorrt'
image_path='mmdetection/demo/demo.jpg'
img = cv2.imread(image_path)
detector = Detector(model_path=model_path,
                    device_name='cuda',
                    device_id=0)
bboxes, labels, _ = detector(img)
```

Why OpenMMLab

Unified architecture	Learn once, use everywhere; implement once, use everywhere
Unified benchmark	Provide fair baselines for academic research
Modular design	Fast to develop and try new components
High-quality Implementation	Efficient, high performance, good code style

Architecture Design

OpenMMLab

Registry

The basis of modular design

Config

Construct modules

Manage experiments

Runner&Hook

Unified training interfaces

Customizable pipelines

Registry

Build an instance with custom configs

1. Register

```python
BACKBONES = Registry('backbones')

@BACKBONES.register_module()
class ResNet(nn.Module):

    pass
```

Registry

BACKBONES

```
'ResNet' -> <class 'ResNet'>
```

2. Build

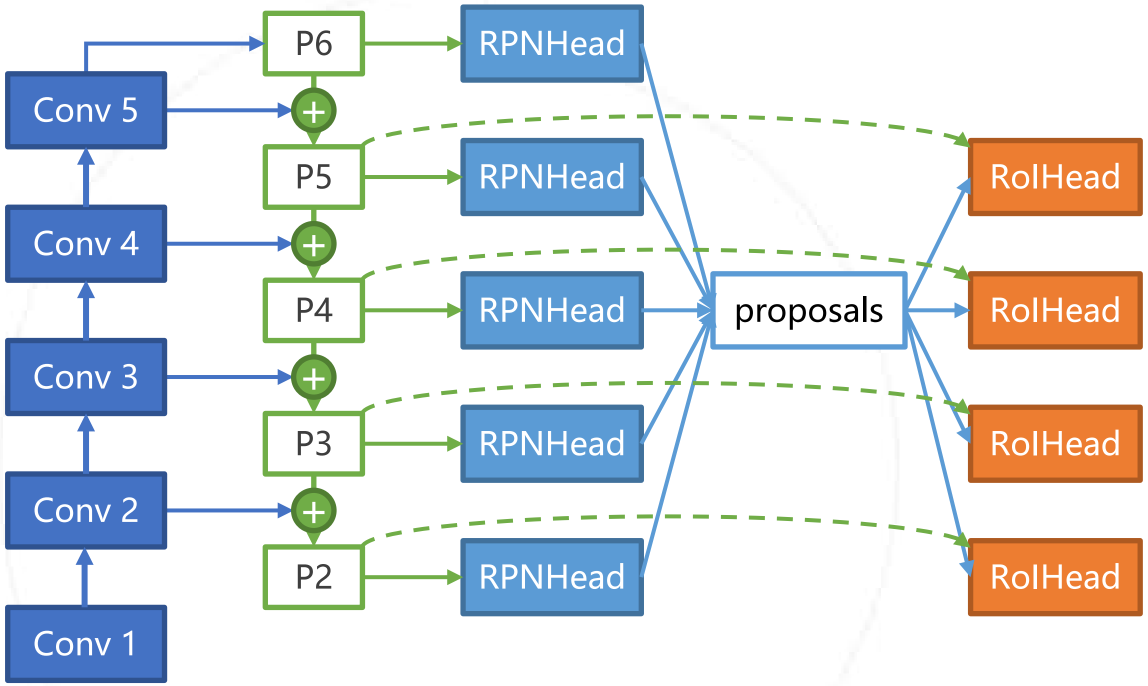
```python
config = dict(type='ResNet')
backbone = build_backbone(config, BACKBONES)
```

Config

Config

```
model = dict(
    type='FasterRCNN',
    pretrained='torchvision://resnet50',
    backbone=dict(
        type='ResNet',
        depth=50,
        ...),
    neck=dict(
        type='FPN',
        ...),
    rpn_head=dict(
        type='RPNHead',
        ...),
    roi_head=dict(
        type='StandardRoIHead',
        bbox_roi_extractor=dict(
            type='SingleRoIExtractor',
            ...),
        bbox_head=dict(
            type='Shared2FCBBoxHead',
            ...))
)
```

Module



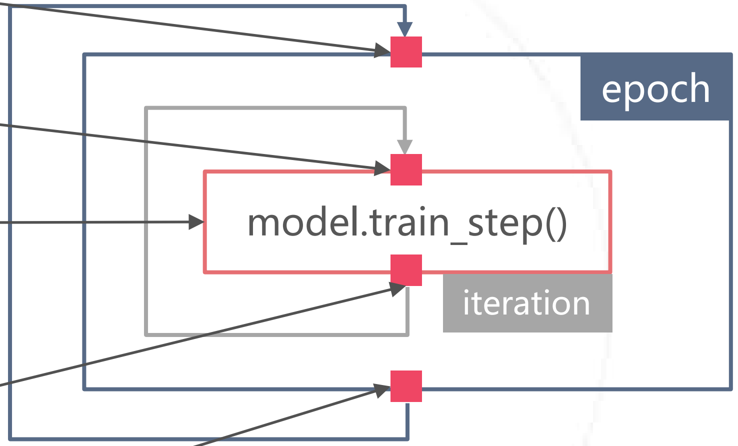
Runner&Hook

```python
model = torch.nn.parallel.DistributedDataParallel(SomeNet(), device_ids=[args.gpu])
optimizer = torch.optim.SGD(...)
train_loader = torch.utils.data.DataLoader(...)

def adjust_learning_rate():
    pass

def record_and_log_loss():
    pass

for epoch in range(args.epochs):
    adjust_learning_rate(optimizer, epoch, args)

    # train for one epoch
    for i, (images, target) in enumerate(train_loader):
        # measure data loading time
        data_time.update(time.time() - end)

        # compute output
        output = model(images)
        loss = criterion(output, target)

        # compute gradient and do SGD step
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # measure accuracy and record loss
        record_and_log_loss(loss)

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

        # print progress
        if i % args.print_freq == 0:
            progress.display(i)

    # evaluate on validation set
    save_checkpoint()
```

ImageNet Example

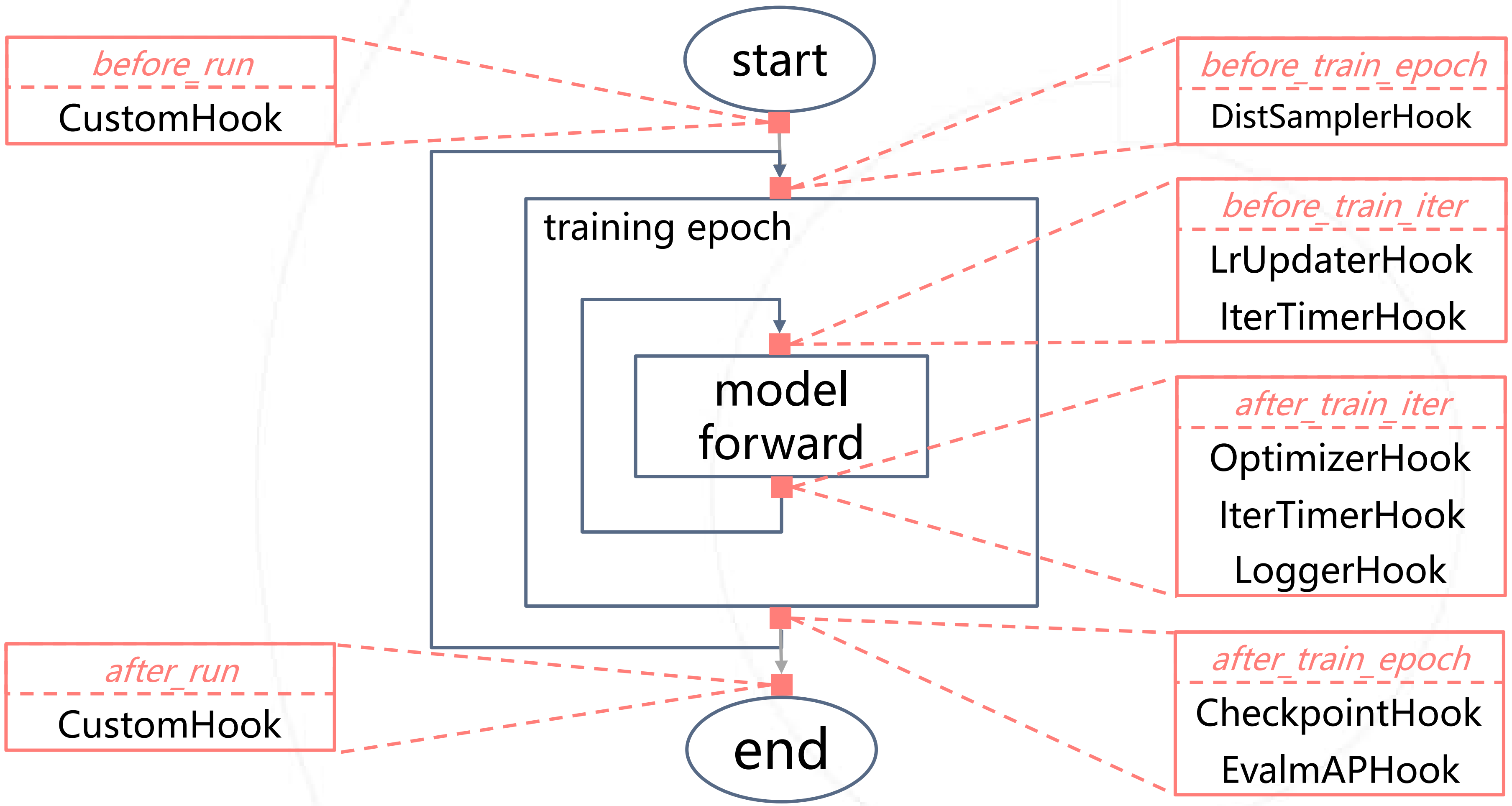
Runner: execution loop and core logic (fetch data and forward model)

Hook: custom logic and facilities (logging, visualization, lr scheduler, checkpointing, etc)



Runner&Hook

Another example



Quick Tour for Developers and Researchers

1. How to support a new dataset
2. How to develop a new model

General Guideline

OpenMMLab

Project	
Maintained in an isolated folder	model.py, data.py, utils.py, config.py, …

Dependencies	
Use pip to install the libraries	mmcv, mmdet, mmcls, …

Best Practice with MIM

Package management

```
pip install openmim>=0.1.1  # install mim through pypi
mim install mmcv-full==1.3.5
mim install mmdet==2.13.0
mim install mmsegmentation=0.14.0
```

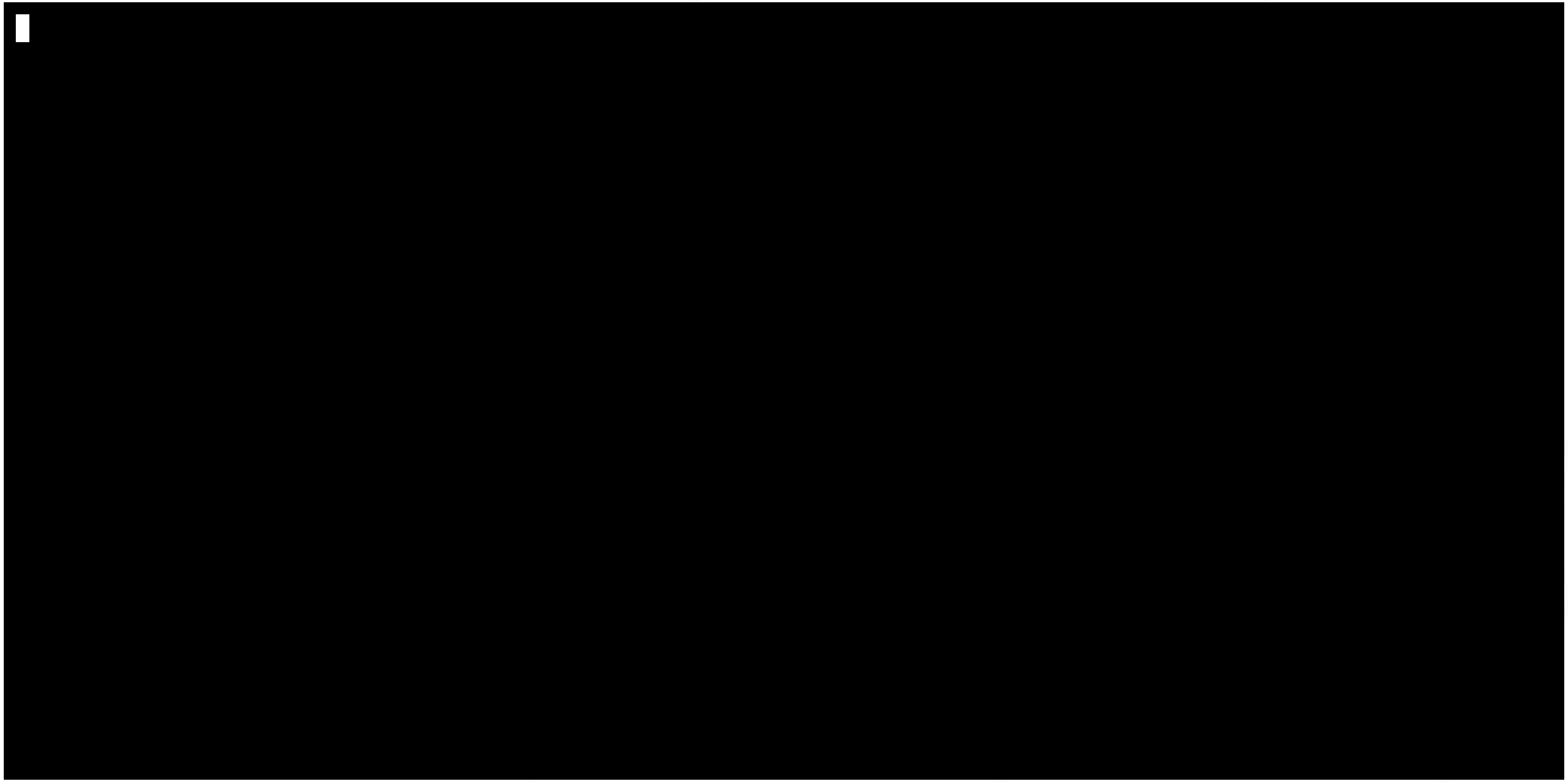
Unified entrypoint for scripts

mim train mmdet → mmdetection/tools/train.py

mim test mmseg → mmsegmentation/tools/test.py

mim train mmcls → mmclassification/tools/train.py

Best Practice with MIM



Find out more at

Support a New Dataset

Process the dataset → Implement a new class → Modify the config file → Train and test

optional

Support a New Dataset

Process the dataset

```
python -u nuim_converter.py \
    --data-root $DATA \
    --versions $VERSIONS \
    --out-dir $OUT \
    --nproc $NUM_WORKERS
```

```
project
├── nuimages
│   ├── annotations
│   │   ├── nuimages_v1.0-mini.json
│   │   ├── nuimages_v1.0-train.json
│   │   ├── nuimages_v1.0-val.json
│   │   ├── nuimages_v1.0-val2400.json
│   │   ├── nuimages_v1.0-test.json
│   │   └── semantic_masks
│   │       ├── xxxxx.png
│   │       └── xxxxx.png
│   └── samples
```

Support a New Dataset

OpenMMLab

Implement a new class

```
project
├── configs
│   ├── _base_
│   │   ├── datasets
│   │   │   └── nuimages.py
│   │   ├── default_runtime.py
│   │   ├── models
│   │   │   └── pspnet_r50-d8.py
│   │   └── schedules
│   │       └── schedule_80k.py
│   └── pspnet
│       └── pspnet_r18-d8_512x1024_80k_nuim.py
├── nuim_converter.py
└── nuim_dataset.py
```

```python
import os.path as osp

import mmcv
from mmcv.utils import print_log
from mmseg.datasets import CustomDataset
from mmseg.datasets.builder import DATASETS
from mmseg.utils import get_root_logger


@DATASETS.register_module()
class NuImagesDataset(CustomDataset):
    CLASSES = ()

    def load_annotations(self, img_dir, img_suffix, ann_dir,
                         seg_map_suffix, split):

        annotations = mmcv.load(split)
        img_infos = []
        for img in annotations['images']:
            img_info = dict(filename=img['file_name'])
            seg_map = img_info['filename'].replace(
                img_suffix, seg_map_suffix)
            img_info['ann'] = dict(
                seg_map=osp.join('semantic_masks', seg_map))
            img_infos.append(img_info)

        print_log(
            f'Loaded {len(img_infos)} images from {ann_dir}',
            logger=get_root_logger())
        return img_infos
```

Use mmseg as a library like PyTorch and torchvision

Register the dataset into the DATASETS registry

Inherits from CustomDataset

Override the load_annotations method

Support a New Dataset

Modify the config file

```
project
├── configs
│   ├── _base_
│   │   ├── datasets
│   │   │   └── nuimages.py
│   │   ├── default_runtime.py
│   │   ├── models
│   │   │   └── pspnet_r50-d8.py
│   │   └── schedules
│   │       └── schedule_80k.py
│   └── pspnet
│       └── pspnet_r18-d8_512x1024_80k_nuim.py
├── nuim_converter.py
└── nuim_dataset.py
```

```python
dataset_type = 'NuImagesDataset'
data_root = 'data/nuimages/'
train_pipeline = [
    ...
]
test_pipeline = [
    ...
]
data = dict(
    samples_per_gpu=2,
    workers_per_gpu=2,
    train=dict(
        type=dataset_type,
        data_root=data_root,
        img_dir='',
        ann_dir='annotations/',
        split='annotations/nuimages_v1.0-train.json',
        pipeline=train_pipeline),
    val=dict(
        type=dataset_type,
        ...),
    test=dict(
        type=dataset_type,
        ...))

custom_imports = dict(
    imports=['nuim_dataset'],
    allow_failed_imports=False)
```

Define data pipeline of the dataset

Make the file imported so that nuImagesDataset can be registered

Support a New Dataset

Train and test

Train the model

```
PYTHONPATH='.'$PYTHONPATH mim train mmseg \
    configs/pspnet/pspnet_r18-d8_512x1024_80k_nuim.py
    --work-dir $WORK_DIR \
    --launcher slurm -G 8 -p $PARTITION
```

Test the trained model

```
PYTHONPATH='.'$PYTHONPATH mim test mmseg \
    configs/pspnet/pspnet_r18-d8_512x1024_80k_nuim.py
    --checkpoint $WORK_DIR/latest.pth \
    --launcher slurm -G 8 -p $PARTITION \
    --eval mIoU
```

Develop a New Model

Implement the model → Modify the config file → Train and test

Develop a New Algorithm

Implement the model

```
├── configs
│    ├── swin_classifier
│    │    └──swin_tiny_224_imagenet.py
│    ├── swin_mask_rcnn
│    │    └──mask_rcnn_swim-t-p4-w7_fpn_1x_coco.py
│    └── swin_upernet
│         └── upernet_swin-t_512x512_160k_8x2_ade20k.py
└── swin
     └── swin_transformer.py
```

```python
from mmcls.models import BACKBONES


@BACKBONES.register_module()
class SwinTransformer(nn.Module):
    # code implementation
    def __init__(self, *args, **kwargs):
        super().__init__()
```

Register

```python
@BACKBONES.register_module()
class SwinTransformer(nn.Module)
```

↓

Registry in MMCls BACKBONES

```
'SwinTransformer' -> <class 'SwinTransformer'>
```

↓

Build

```python
module_cfg = dict(type='SwinTransformer')
module = build_backbone(module_cfg)
```

Develop a New Algorithm

OpenMMLab

Modify the config file

```
├── configs
|    ├── swin_classifier
|    |    └──swin_tiny_224_imagenet.py
|    ├── swin_mask_rcnn
|    |    └──mask_rcnn_swim-t-p4-w7_fpn_1x_coco.py
|    └── swin_upernet
|         └── upernet_swin-t_512x512_160k_8x2_ade20k.py
└── swin
     └── swin_transformer.py
```

```python
_base_ = [
    '../_base_/datasets/imagenet_bs128_swin_224.py',
    '../_base_/schedules/imagenet_bs1024_adamw_swin.py',
    '../_base_/default_runtime.py'
]

model = dict(
    type='ImageClassifier',
    backbone=dict(
        type='SwinTransformer', arch='tiny', img_size=224, drop_path_rate=0.2),
    neck=dict(type='GlobalAveragePooling', dim=1),
    head=dict(
        type='SwinLinearClsHead',
        num_classes=1000,
        in_channels=768,
        loss=dict(type='CrossEntropyLoss', use_soft=True),
        cal_acc=False),
    train_cfg=dict(
        cutmixup=dict(
            mixup_alpha=0.8,
            cutmix_alpha=1.0,
            prob=1.0,
            switch_prob=0.5,
            mode='batch',
            label_smoothing=0.1)))

custom_imports = dict(
    imports=['swin.swin_transformer'], allow_failed_imports=False)
```

Develop a New Algorithm

OpenMMLab

Modify the config file

```
├── configs
│    ├── swin_classifier
│    │    └──swin_tiny_224_imagenet.py
│    ├── swin_mask_rcnn
│    │    └──mask_rcnn_swim-t-p4-w7_fpn_1x_coco.py
│    └── swin_upernet
│         └── upernet_swin-t_512x512_160k_8x2_ade20k.py
└── swin
     └── swin_transformer.py
```

```
_base_ = [
    '../_base_/models/mask_rcnn_r50_fpn.py',
    '../_base_/datasets/coco_instance.py',
    '../_base_/schedules/schedule_1x.py',
    '../_base_/default_runtime_det.py'
]

model = dict(
    pretrained='./pretrain/swin/swin_tiny_patch4_window7_224.pth',
    backbone=dict(type='mmcls.SwinTransformer'))

custom_imports = dict(
    imports=['swin.swin_transformer'], allow_failed_imports=False)
```

Develop a New Algorithm

Train and test

Train Swin Mask-RCNN with MMDetection

```
PYTHONPATH='.':$PYTHONPATH mim train mmdet \
    configs/swin_mask_rcnn/mask_rcnn_swim-t-p4-w7_fpn_fp16_1x_coco.py \
    --work-dir ../work_dir/mask_rcnn_swim-t-p4-w7_fpn_fp16_1x_coco.py \
    --launcher slurm --partition $PARTITION -G 8 --gpus-per-node 8 \
    --srun-args $SRUN_ARGS
```

Find out more at

Resources

OpenMMLab

Homepage
https://openmmlab.com/

GitHub
https://github.com/open-mmlab

Twitter
@OpenMMLab

Thank you!