MMTracking

发布 *0.14.0*

MMTracking Authors

2022 年 10 月 21 日

开始你的第一步

您可以在页面左下角切换中英文文档。

CHAPTER 1

依赖

- Linux | macOS | Windows
- Python 3.6+
- PyTorch 1.3+
- CUDA 9.2+ (如果您从源代码构建 PyTorch, 那么 CUDA9.0 也是兼容的)
- GCC 5+
- MMCV
- MMDetection

兼容的 MMTracking，MMCV 和 MMDetection 版本如下，请安装正确的版本以避免安装问题。

CHAPTER 2

安装

2.1 详细说明

1. 创建一个 conda 虚拟环境并激活它：

```
conda create -n open-mmlab python=3.7 -y
conda activate open-mmlab
```

2. 按照 PyTorch 官网安装 PyTorch 和 torchvision，例如：

```
conda install pytorch torchvision -c pytorch
```

注意：请确保 CUDA 编译版本和运行版本匹配。可以在 PyTorch 官网查看预编译包所支持的 CUDA 版本。

例 1 例如在 `/usr/local/cuda` 安装了 CUDA 10.1，并想安装 PyTorch 1.5，则需要安装支持 CUDA 10.1 的预构建 PyTorch：

```
conda install pytorch==1.5 cudatoolkit=10.1 torchvision -c pytorch
```

例 2 例如在 `/usr/local/cuda` 安装了 CUDA 9.2，并想安装 PyTorch 1.3.1，则需要安装支持 CUDA 9.2 的预构建 PyTorch：

```
conda install pytorch=1.3.1 cudatoolkit=9.2 torchvision=0.4.2 -c pytorch
```

如果不是安装预构建的包，而是从源码中构建 PyTorch，则可以使用更多的 CUDA 版本，例如 CUDA 9.0。

3. 为 VOT 数据集测试评估安装额外库（可选）

   如果您想要在 VOT Challenge 上进行评估，请在安装 mmcv 与 mmdetection 之前安装 vot-toolkit ，以避免可能出现的一些依赖间的 numpy 版本要求冲突。

```
pip install git+https://github.com/votchallenge/toolkit.git
```

4. 安装 mmcv-full，我们推荐您安装以下预构建包：

```
# pip install mmcv-full -f https://download.openmmlab.com/mmcv/dist/{cu_version}/
↪{torch_version}/index.html
pip install mmcv-full -f https://download.openmmlab.com/mmcv/dist/cu102/torch1.10.
↪0/index.html
```

   PyTorch 在 1.x.0 和 1.x.1 之间通常是兼容的，故 mmcv-full 只提供 1.x.0 的编译包。如果你的 PyTorch 版本是 1.x.1，你可以放心地安装在 1.x.0 版本编译的 mmcv-full。

```
# 我们可以忽略 PyTorch 的小版本号
pip install mmcv-full -f https://download.openmmlab.com/mmcv/dist/cu102/torch1.10/
↪index.html
```

   请参阅 MMCV 了解不同版本的 MMCV 与不同版本的 PyTorch 和 CUDA 的兼容情况。同时，您可以使用以下命令从源码编译 MMCV：

```
git clone https://github.com/open-mmlab/mmcv.git
cd mmcv
MMCV_WITH_OPS=1 pip install -e .  # package mmcv-full will be installed after␣
↪this step
cd ..
```

5. 安装 MMDetection：

```
pip install mmdet
```

   如果您想修改代码，也可以从源码构建 MMDetection：

```
git clone https://github.com/open-mmlab/mmdetection.git
cd mmdetection
pip install -r requirements/build.txt
pip install -v -e .  # or "python setup.py develop"
```

6. 将 MMTracking 仓库克隆到本地：

```
git clone https://github.com/open-mmlab/mmtracking.git
cd mmtracking
```

7. 首先安装依赖，然后安装 MMTracking：

```
pip install -r requirements/build.txt
pip install -v -e .  # or "python setup.py develop"
```

8. 安装额外的依赖：

- 为 MOTChallenge 评估：

```
pip install git+https://github.com/JonathonLuiten/TrackEval.git
```

- 为 LVIS 评估：

```
pip install git+https://github.com/lvis-dataset/lvis-api.git
```

- 为 TAO 评估：

```
pip install git+https://github.com/TAO-Dataset/tao.git
```

注意：

(1) 按照上述说明，MMTracking 将以 `dev` 模式安装，因此在本地对代码做的任何修改都会生效，无需重新安装；

(2) 如果希望使用 `opencv-python-headless` 而不是 `opencv-python`，可以在安装 MMCV 之前安装；

2.2 从零开始设置脚本

假设当前已经成功安装 CUDA 10.1，这里提供了一个完整的基于 conda 安装 MMTracking 的脚本：

```
conda create -n open-mmlab python=3.7 -y
conda activate open-mmlab

conda install pytorch==1.6.0 torchvision==0.7.0 cudatoolkit=10.1 -c pytorch -y

pip install git+https://github.com/votchallenge/toolkit.git (可选)

# 安装最新版本的 mmcv
pip install mmcv-full -f https://download.openmmlab.com/mmcv/dist/cu101/torch1.6.0/
↪index.html

# 安装 mmdetection
```

(下页继续)

(续上页)

```
pip install mmdet

# 安装 mmtracking
git clone https://github.com/open-mmlab/mmtracking.git
cd mmtracking
pip install -r requirements/build.txt
pip install -v -e .
pip install git+https://github.com/JonathonLuiten/TrackEval.git
pip install git+https://github.com/lvis-dataset/lvis-api.git
pip install git+https://github.com/TAO-Dataset/tao.git
```

2.3 使用多个 MMTracking 版本进行开发

训练和测试脚本已经在 PYTHONPATH 中进行了修改，以确保脚本使用当前目录中的 MMTracking。

要使用环境中默认安装的 MMTracking 而不是当前正在使用的版本，可以删除出现在相关脚本中的代码：

```
PYTHONPATH="$(dirname $0)/..":$PYTHONPATH
```

2.4 验证

为了验证是否正确安装了 MMTracking 和所需的环境，我们可以运行 MOT、VID、SOT 的示例脚本。

运行 MOT 演示脚本您可以看到输出一个命名为 mot.mp4 的视频文件：

```
python demo/demo_mot_vis.py configs/mot/deepsort/sort_faster-rcnn_fpn_4e_mot17-
↪private.py --input demo/demo.mp4 --output mot.mp4
```

CHAPTER 3

模型库统计数据

- 论文数量: 12
  - ABSTRACT: 12
- 模型数量: 44
  - [ABSTRACT] ByteTrack: Multi-Object Tracking by Associating Every Detection Box (2 ckpts)
  - [ABSTRACT] Simple online and realtime tracking with a deep association metric (2 ckpts)
  - [ABSTRACT] Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking (1 ckpts)
  - [ABSTRACT] Quasi-Dense Similarity Learning for Multiple Object Tracking (4 ckpts)
  - [ABSTRACT] Tracking without Bells and Whistles (7 ckpts)
  - [ABSTRACT] Siamrpn++: Evolution of Siamese Visual Tracking With Very Deep Networks (5 ckpts)
  - [ABSTRACT] Learning Spatio-Temporal Transformer for Visual Tracking (4 ckpts)
  - [ABSTRACT] Deep Feature Flow for Video Recognition (3 ckpts)
  - [ABSTRACT] Flow-guided Feature Aggregation for Video Object Detection (3 ckpts)
  - [ABSTRACT] Sequence Level Semantics Aggregation for Video Object Detection (4 ckpts)
  - [ABSTRACT] Temporal RoI Align for Video Object Recognition (3 ckpts)
  - [ABSTRACT] Video Instance Segmentation (6 ckpts)

CHAPTER 4

基准测试与模型库

4.1 通用设置

- 我们默认使用分布式训练。
- 所有 pytorch 类型的预训练骨干网络都是来自 Pytorch 的模型库。
- 为了与其他代码库进行公平比较，我们以全部 8 个 GPU 的 `torch.cuda.max_memory_allocated()` 的最大值作为 GPU 显存使用量。请注意，此值通常小于 `nvidia-smi` 显示的值。
- 该推理时间不包含数据加载时间，推理时间结果是通过脚本 `tools/analysis/benchmark.py` 获得的，该脚本计算处理 2000 张图像的平均时间。
- 速度基准测试的环境如下：

  硬件环境：

  - 8 NVIDIA Tesla V100 (32G) GPUs
  - Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz

  软件环境：

  - Python 3.7
  - PyTorch 1.5
  - CUDA 10.1
  - CUDNN 7.6.03

- NCCL 2.4.08

4.2 视频目标检测基线

4.2.1 DFF (CVPR 2017)

详情请参考 DFF。

4.2.2 FGFA (ICCV 2017)

详情请参考 FGFA。

4.2.3 SELSA (ICCV 2019)

详情请参考 SELSA。

4.2.4 Temporal RoI Align (AAAI 2021)

详情请参考 Temporal RoI Align。

4.3 多目标跟踪基线

4.3.1 SORT/DeepSORT (ICIP 2016/2017)

详情请参考 SORT/DeepSORT。

4.3.2 Tracktor (ICCV 2019)

详情请参考 Tracktor。

4.3.3 QDTrack (CVPR 2021)

详情请参考 QDTrack。

4.3.4 ByteTrack (ECCV 2022)

详情请参考 ByteTrack。

4.3.5 OC-SORT (ArXiv 2022)

详情请参考 OC-SORT。

4.4 单目标跟踪基线

4.4.1 SiameseRPN++ (CVPR 2019)

详情请参考 SiameseRPN++。

4.4.2 STARK (ICCV 2021)

详情请参考 STARK。

4.5 视频个例分割基线

4.5.1 MaskTrack R-CNN (ICCV 2019)

详情请参考 MaskTrack R-CNN。

CHAPTER 5

数据集准备

本页提供关于现有基准测试的数据集准备的说明，包括：

- 视频目标检测
  - ILSVRC
- 多目标跟踪
  - MOT Challenge
  - CrowdHuman
  - LVIS
  - TAO
  - DanceTrack
- 单目标跟踪
  - LaSOT
  - UAV123
  - TrackingNet
  - OTB100
  - GOT10k
  - VOT2018
- 视频实例分割

  - YouTube-VIS

5.1 1. 下载数据集

请从官方网站下载数据集。建议将数据集的根目录符号链接到 `$MMTRACKING/data`。

5.1.1 1.1 视频目标检测

- 对于视频目标检测任务的训练和测试，只需要 ILSVRC 数据集。
- `ILSVRC` 下的 `Lists` 包含来自在这里的 txt 文件。

5.1.2 1.2 多目标跟踪

- 对于多目标跟踪任务的训练和测试，需要 MOT Challenge 中的任意一个数据集（比如 MOT17, TAO 和 DanceTrack），CrowdHuman 和 LVIS 可以作为补充数据。
- `tao` 文件夹下包含官方标注的 `annotations` 可以从这里获取。
- `lvis` 文件夹下包含 lvis-v0.5 官方标注的 `annotations` 可以从这里下载。`./tools/convert_datasets/tao/merge_coco_with_lvis.py` 脚本中需到的同义词映射文件 `coco_to_lvis_synset.json` 可以从这里获取。

5.1.3 1.3 单目标跟踪

- 对于单目标跟踪任务的训练和测试，需要 MSCOCO，ILSVRC, LaSOT, UAV123, TrackingNet, OTB100 和 GOT10k 数据集。
- 对于 OTB100 数据集，你不必要手工地从官网下载数据。我们提供了下载脚本。

```
# 通过网页爬虫下载 OTB100 数据集
python ./tools/convert_datasets/otb100/download_otb100.py -o ./data/otb100/zips -p 8
```

- 对于 VOT2018, 我们使用官方的下载脚本。

```
# 通过网页爬虫下载 VOT2018 数据集
python ./tools/convert_datasets/vot/download_vot.py --dataset vot2018 --save_path ./
↪data/vot2018/data
```

5.1.4 1.4 视频实例分割

- 对于视频实例分割任务的训练和测试，只需要 YouTube-VIS 中的任意一个数据集（比如 YouTube-VIS 2019）。

5.1.5 1.5 数据集文件夹结构

如果您的文件夹结构与以下不同，您可能需要更改配置文件中的相应路径。

```
mmtracking
├── mmtrack
├── tools
├── configs
├── data
|   ├── coco
|   |   ├── train2017
|   |   ├── val2017
|   |   ├── test2017
|   |   ├── annotations
|   |
|   ├── ILSVRC
|   |   ├── Data
|   |   |   ├── DET
|   |   |   |   ├── train
|   |   |   |   ├── val
|   |   |   |   ├── test
|   |   |   ├── VID
|   |   |   |   ├── train
|   |   |   |   ├── val
|   |   |   |   ├── test
|   |   ├── Annotations
|   |   |   ├── DET
|   |   |   |   ├── train
|   |   |   |   ├── val
|   |   |   ├── VID
|   |   |   |   ├── train
|   |   |   |   ├── val
|   |   ├── Lists
|   |
|   ├── MOT15/MOT16/MOT17/MOT20
|   |   ├── train
|   |   ├── test
|   |
|   ├── DanceTrack
```

(下页继续)

(续上页)

```
|   |   ├── train
|   |   ├── val
|   |   ├── test
|   |
|   ├── crowdhuman
|   |   ├── annotation_train.odgt
|   |   ├── annotation_val.odgt
|   |   ├── train
|   |   |   ├── Images
|   |   |   ├── CrowdHuman_train01.zip
|   |   |   ├── CrowdHuman_train02.zip
|   |   |   ├── CrowdHuman_train03.zip
|   |   ├── val
|   |   |   ├── Images
|   |   |   ├── CrowdHuman_val.zip
|   |
|   ├── lvis
|   |   ├── train (the same as coco/train2017)
|   |   ├── val (the same as coco/val2017)
|   |   ├── test (the same as coco/test2017)
|   |   ├── annotations
|   |   |   ├── coco_to_lvis_synset.json
|   |   |   ├── lvis_v0.5_train.json
|   |   |   ├── lvis_v0.5_val.json
|   |   |   ├── lvis_v1_train.json
|   |   |   ├── lvis_v1_val.json
|   |   |   ├── lvis_v1_image_info_test_challenge.json
|   |   |   ├── lvis_v1_image_info_test_dev.json
|   |
|   ├── tao
|   |   ├── annotations
|   |   |   ├── test_without_annotations.json
|   |   |   ├── train.json
|   |   |   ├── validation.json
|   |   |   ├── ......
|   |   ├── test
|   |   |   ├── ArgoVerse
|   |   |   ├── AVA
|   |   |   ├── BDD
|   |   |   ├── Charades
|   |   |   ├── HACS
|   |   |   ├── LaSOT
|   |   |   ├── YFCC100M
```

(下页继续)

(续上页)

```
|   |   ├── train
|   |   ├── val
|   |
|   ├── lasot
|   |   ├── LaSOTBenchmark
|   |   |   ├── airplane
|   |   |   |   ├── airplane-1
|   |   |   |   ├── airplane-2
|   |   |   |   ├── ......
|   |   |   ├── ......
|   |
|   ├── UAV123
|   |   ├── data_seq
|   |   |   ├── UAV123
|   |   |   |   ├── bike1
|   |   |   |   ├── boat1
|   |   |   |   ├── ......
|   |   ├── anno
|   |   |   ├── UAV123
|   |
|   ├── trackingnet
|   |   ├── TEST.zip
|   |   ├── TRAIN_0.zip
|   |   ├── ......
|   |   ├── TRAIN_11.zip
|   |
|   ├── otb100
|   |   |── zips
|   |   |   |── Basketball.zip
|   |   |   |── Biker.zip
|   |   |   |──
|   |
|   ├── got10k
|   |   |── full_data
|   |   |   |── train_data
|   |   |   |   ├── GOT-10k_Train_split_01.zip
|   |   |   |   ├── ......
|   |   |   |   ├── GOT-10k_Train_split_19.zip
|   |   |   |   ├── list.txt
|   |   |   |── test_data.zip
|   |   |   |── val_data.zip
|   |
|   ├── vot2018
```

(下页继续)

(续上页)

```
|   |   ├── data
|   |   |   ├── ants1
|   |   |   |   ├──color
|   |
|   ├── youtube_vis_2019
|   |   |── train
|   |   |   |── JPEGImages
|   |   |   |── ......
|   |   |── valid
|   |   |   |── JPEGImages
|   |   |   |── ......
|   |   |── test
|   |   |   |── JPEGImages
|   |   |   |── ......
|   |   |── train.json (the official annotation files)
|   |   |── valid.json (the official annotation files)
|   |   |── test.json (the official annotation files)
|   |
|   ├── youtube_vis_2021
|   |   |── train
|   |   |   |── JPEGImages
|   |   |   |── instances.json (the official annotation files)
|   |   |   |── ......
|   |   |── valid
|   |   |   |── JPEGImages
|   |   |   |── instances.json (the official annotation files)
|   |   |   |── ......
|   |   |── test
|   |   |   |── JPEGImages
|   |   |   |── instances.json (the official annotation files)
|   |   |   |── ......
```

5.2 2. 转换标注格式

我们使用 CocoVID 来维护代码库中所有的数据集。

基于此，您需要将官方的标注转换为此种格式。我们提供的脚本以及用法如下：

```
# ImageNet DET
python ./tools/convert_datasets/ilsvrc/imagenet2coco_det.py -i ./data/ILSVRC -o ./
↪data/ILSVRC/annotations
```

(下页继续)

(续上页)

```
# ImageNet VID
python ./tools/convert_datasets/ilsvrc/imagenet2coco_vid.py -i ./data/ILSVRC -o ./
↪data/ILSVRC/annotations

# MOT17
# MOT Challenge 中其余数据集的处理与 MOT17 相同
python ./tools/convert_datasets/mot/mot2coco.py -i ./data/MOT17/ -o ./data/MOT17/
↪annotations --split-train --convert-det
python ./tools/convert_datasets/mot/mot2reid.py -i ./data/MOT17/ -o ./data/MOT17/reid␣
↪--val-split 0.2 --vis-threshold 0.3

# DanceTrack
python ./tools/convert_datasets/dancetrack/dancetrack2coco.py -i ./data/DanceTrack ./
↪data/DanceTrack/annotations

# CrowdHuman
python ./tools/convert_datasets/mot/crowdhuman2coco.py -i ./data/crowdhuman -o ./data/
↪crowdhuman/annotations

# LVIS
# 合并 LVIS 和 COCO 的标注来训练 QDTrack
python ./tools/convert_datasets/tao/merge_coco_with_lvis.py --lvis ./data/lvis/
↪annotations/lvis_v0.5_train.json --coco ./data/coco/annotations/instances_train2017.
↪json --mapping ./data/lvis/annotations/coco_to_lvis_synset.json --output-json ./
↪data/lvis/annotations/lvisv0.5+coco_train.json

# TAO
# 为 QDTrack 生成过滤后的 json 文件
python ./tools/convert_datasets/tao/tao2coco.py -i ./data/tao/annotations --filter-
↪classes

# LaSOT
python ./tools/convert_datasets/lasot/gen_lasot_infos.py -i ./data/lasot/
↪LaSOTBenchmark -o ./data/lasot/annotations

# UAV123
# 下载标注
# 由于 UAV123 数据集的所有视频的标注信息不具有统一性，我们仅需下载提前生成的数据信息文件即可。
wget https://download.openmmlab.com/mmtracking/data/uav123_infos.txt -P data/uav123/
↪annotations

# TrackingNet
# 解压目录 'data/trackingnet/' 下的所有 '*.zip' 文件
```

(下页继续)

(续上页)

```
bash ./tools/convert_datasets/trackingnet/unzip_trackingnet.sh ./data/trackingnet
# 生成标注
python ./tools/convert_datasets/trackingnet/gen_trackingnet_infos.py -i ./data/
↪trackingnet -o ./data/trackingnet/annotations

# OTB100
# 解压目录 'data/otb100/zips' 下的所有 '*.zip' 文件
bash ./tools/convert_datasets/otb100/unzip_otb100.sh ./data/otb100
# 下载标注
# 由于 OTB100 数据集的所有视频的标注信息不具有统一性，我们仅需下载提前生成的数据信息文件即可。
wget https://download.openmmlab.com/mmtracking/data/otb100_infos.txt -P data/otb100/
↪annotations

# GOT10k
# 解压 'data/got10k/full_data/test_data.zip', 'data/got10k/full_data/val_data.zip' 和 目
录'data/got10k/full_data/train_data/' 下的所有 '*.zip' 文件
bash ./tools/convert_datasets/got10k/unzip_got10k.sh ./data/got10k
# 生成标注
python ./tools/convert_datasets/got10k/gen_got10k_infos.py -i ./data/got10k -o ./data/
↪got10k/annotations

# VOT2018
python ./tools/convert_datasets/vot/gen_vot_infos.py -i ./data/vot2018 -o ./data/
↪vot2018/annotations --dataset_type vot2018

# YouTube-VIS 2019
python ./tools/convert_datasets/youtubevis/youtubevis2coco.py -i ./data/youtube_vis_
↪2019 -o ./data/youtube_vis_2019/annotations --version 2019

# YouTube-VIS 2021
python ./tools/convert_datasets/youtubevis/youtubevis2coco.py -i ./data/youtube_vis_
↪2021 -o ./data/youtube_vis_2021/annotations --version 2021
```

完成以上格式转换后，文件目录结构如下：

```
mmtracking
├── mmtrack
├── tools
├── configs
├── data
|   ├── coco
|   |   ├── train2017
|   |   ├── val2017
|   |   ├── test2017
```

(下页继续)

(续上页)

```
|   |   ├── annotations
|   |
|   ├── ILSVRC
|   |   ├── Data
|   |   |   ├── DET
|   |   |   |   ├── train
|   |   |   |   ├── val
|   |   |   |   ├── test
|   |   |   ├── VID
|   |   |   |   ├── train
|   |   |   |   ├── val
|   |   |   |   ├── test
|   |   ├── Annotations (the official annotation files)
|   |   |   ├── DET
|   |   |   |   ├── train
|   |   |   |   ├── val
|   |   |   ├── VID
|   |   |   |   ├── train
|   |   |   |   ├── val
|   |   ├── Lists
|   |   ├── annotations (the converted annotation files)
|   |
|   ├── MOT15/MOT16/MOT17/MOT20
|   |   ├── train
|   |   ├── test
|   |   ├── annotations
|   |   ├── reid
|   |   |   ├── imgs
|   |   |   ├── meta
|   |
|   ├── DanceTrack
|   |   ├── train
|   |   ├── val
|   |   ├── test
|   |   ├── annotations
|   |
|   ├── crowdhuman
|   |   ├── annotation_train.odgt
|   |   ├── annotation_val.odgt
|   |   ├── train
|   |   |   ├── Images
|   |   |   ├── CrowdHuman_train01.zip
|   |   |   ├── CrowdHuman_train02.zip
```

(下页继续)

(续上页)

```
|   |   |   ├── CrowdHuman_train03.zip
|   |   ├── val
|   |   |   ├── Images
|   |   |   ├── CrowdHuman_val.zip
|   |   ├── annotations
|   |   |   ├── crowdhuman_train.json
|   |   |   ├── crowdhuman_val.json
|   |
|   ├── lvis
|   |   ├── train (the same as coco/train2017)
|   |   ├── val (the same as coco/val2017)
|   |   ├── test (the same as coco/test2017)
|   |   ├── annotations
|   |   |   ├── coco_to_lvis_synset.json
|   |   |   ├── lvisv0.5+coco_train.json
|   |   |   ├── lvis_v0.5_train.json
|   |   |   ├── lvis_v0.5_val.json
|   |   |   ├── lvis_v1_train.json
|   |   |   ├── lvis_v1_val.json
|   |   |   ├── lvis_v1_image_info_test_challenge.json
|   |   |   ├── lvis_v1_image_info_test_dev.json
|   |
|   ├── tao
|   |   ├── annotations
|   |   |   ├── test_482_classes.json
|   |   |   ├── test_without_annotations.json
|   |   |   ├── train.json
|   |   |   ├── train_482_classes.json
|   |   |   ├── validation.json
|   |   |   ├── validation_482_classes.json
|   |   |   ├── ......
|   |   ├── test
|   |   |   ├── ArgoVerse
|   |   |   ├── AVA
|   |   |   ├── BDD
|   |   |   ├── Charades
|   |   |   ├── HACS
|   |   |   ├── LaSOT
|   |   |   ├── YFCC100M
|   |   ├── train
|   |   ├── val
|   |
|   ├── lasot
```

(下页继续)

(续上页)

```
|   |   ├── LaSOTBenchmark
|   |   |   ├── airplane
|   |   |   |   ├── airplane-1
|   |   |   |   ├── airplane-2
|   |   |   |   ├── ......
|   |   |   ├── ......
|   |   ├── annotations
|   |
|   ├── UAV123
|   |   ├── data_seq
|   |   |   ├── UAV123
|   |   |   |   ├── bike1
|   |   |   |   ├── boat1
|   |   |   |   ├── ......
|   |   ├── anno (the official annotation files)
|   |   |   ├── UAV123
|   |   ├── annotations (the converted annotation file)
|   |
|   ├── trackingnet
|   |   ├── TEST
|   |   |   ├── anno (the official annotation files)
|   |   |   ├── zips
|   |   |   ├── frames (the unzipped folders)
|   |   |   |   ├── 0-6LB4FqxoE_0
|   |   |   |   ├── 07Ysk1C0ZX0_0
|   |   |   |   ├── ......
|   |   ├── TRAIN_0
|   |   |   ├── anno (the official annotation files)
|   |   |   ├── zips
|   |   |   ├── frames (the unzipped folders)
|   |   |   |   ├── -3TIfnTSM6c_2
|   |   |   |   ├── a1qoB1eERn0_0
|   |   |   |   ├── ......
|   |   ├── ......
|   |   ├── TRAIN_11
|   |   ├── annotations (the converted annotation file)
|   |
|   ├── otb100
|   |   ├── zips
|   |   |   ├── Basketball.zip
|   |   |   ├── Biker.zip
|   |   |   |── ......
|   |   ├── annotations
```

(下页继续)

(续上页)

```
|   |   ├── data
|   |   |   ├── Basketball
|   |   |   |   ├── img
|   |   |   ├── ......
|   |
|   ├── got10k
|   |   |── full_data
|   |   |   |── train_data
|   |   |   |   ├── GOT-10k_Train_split_01.zip
|   |   |   |   ├── ......
|   |   |   |   ├── GOT-10k_Train_split_19.zip
|   |   |   |   ├── list.txt
|   |   |   |── test_data.zip
|   |   |   |── val_data.zip
|   |   |── train
|   |   |   ├── GOT-10k_Train_000001
|   |   |   |   ├── ......
|   |   |   ├── GOT-10k_Train_009335
|   |   |   ├── list.txt
|   |   |── test
|   |   |   ├── GOT-10k_Test_000001
|   |   |   |   ├── ......
|   |   |   ├── GOT-10k_Test_000180
|   |   |   ├── list.txt
|   |   |── val
|   |   |   ├── GOT-10k_Val_000001
|   |   |   |   ├── ......
|   |   |   ├── GOT-10k_Val_000180
|   |   |   ├── list.txt
|   |   |── annotations
|   |
|   ├── vot2018
|   |   ├── data
|   |   |   ├── ants1
|   |   |   |   ├──color
|   |   ├── annotations
|   |   |   ├── ......
|   |
|   ├── youtube_vis_2019
|   |   |── train
|   |   |   |── JPEGImages
|   |   |   |── ......
|   |   |── valid
```

(下页继续)

(续上页)

```
|   |   |   |── JPEGImages
|   |   |   |── ......
|   |   |── test
|   |   |   |── JPEGImages
|   |   |   |── ......
|   |   |── train.json (the official annotation files)
|   |   |── valid.json (the official annotation files)
|   |   |── test.json (the official annotation files)
|   |   |── annotations (the converted annotation file)
|   |
|   ├── youtube_vis_2021
|   |   |── train
|   |   |   |── JPEGImages
|   |   |   |── instances.json (the official annotation files)
|   |   |   |── ......
|   |   |── valid
|   |   |   |── JPEGImages
|   |   |   |── instances.json (the official annotation files)
|   |   |   |── ......
|   |   |── test
|   |   |   |── JPEGImages
|   |   |   |── instances.json (the official annotation files)
|   |   |   |── ......
|   |   |── annotations (the converted annotation file)
```

5.2.1 ILSVRC 的标注文件夹

在 `data/ILSVRC/annotations` 中有 3 个 JSON 文件:

`imagenet_det_30plus1cls.json`: 包含 ImageNet DET 训练集标注信息的 json 文件。`30plus1cls` 中的 `30` 表示本数据集与 ImageNet VID 数据集重合的 30 类，`1cls` 表示我们将 ImageNet Det 数据集中的其余 170 类作为一类，并命名为 `other_categeries`。

`imagenet_vid_train.json`: 包含 ImageNet VID 训练集标注信息的 JSON 文件。

`imagenet_vid_val.json`: 包含 ImageNet VID 验证集标注信息的 JSON 文件。

5.2.2 MOT15/MOT16/MOT17/MOT20 的标注和 reid 文件夹

我们以 MOT17 为例，其余数据集结构相似。

在 data/MOT17/annotations 中有 8 个 JSON 文件:

train_cocoformat.json: 包含 MOT17 训练集标注信息的 JSON 文件。

train_detections.pkl: 包含 MOT17 训练集公共检测结果信息的 pickle 文件。

test_cocoformat.json: 包含 MOT17 测试集标注信息的 JSON 文件。

test_detections.pkl: 包含 MOT17 测试集公共检测结果信息的 pickle 文件。

half-train_cocoformat.json, half-train_detections.pkl, half-val_cocoformat.json 以及 half-val_detections.pkl 具有和 train_cocoformat.json、train_detections.pkl 相似的含义。half 意味着我们将训练集中的每个视频分成两半。前一半标记为 half-train, 后一半标记为 half-val。

data/MOT17/reid 的目录结构如下:

```
reid
├── imgs
|   ├── MOT17-02-FRCNN_000002
|   |   ├── 000000.jpg
|   |   ├── 000001.jpg
|   |   ├── ......
|   ├── MOT17-02-FRCNN_000003
|   |   ├── 000000.jpg
|   |   ├── 000001.jpg
|   |   ├── ......
├── meta
|   ├── train_80.txt
|   ├── val_20.txt
```

train_80.txt 中的 80 意味着将全部 ReID 数据集的 80% 作为训练集，剩余的 20% 作为验证集。

训练集标注 train_80.txt 中每一行包含一个文件名和其对应的图片物体真实标签。格式如下:

```
MOT17-05-FRCNN_000110/000018.jpg 0
MOT17-13-FRCNN_000146/000014.jpg 1
MOT17-05-FRCNN_000088/000004.jpg 2
MOT17-02-FRCNN_000009/000081.jpg 3
```

MOT17-05-FRCNN_000110 表示 MOT17-05-FRCNN 视频中的第 110 个人。

验证集标注 val_20.txt 的结构和上面类似。

reid/imgs 中的图片是从 MOT17/train 中原始图片根据对应的 gt.txt 裁剪得到。真实类别标签值在 [0, num_classes - 1] 范围内。

5.2.3 crowdhuman 的标注文件夹

在 data/crowdhuman/annotations 中有 2 个 JSON 文件:

crowdhuman_train.json: 包含 CrowdHuman 训练集标注信息的 JSON 文件。crowdhuman_val.json: 包含 CrowdHuman 验证集标注信息的 JSON 文件。

5.2.4 lvis 的标注文件夹

在 data/lvis/annotations 中有 8 个 JSON 文件:

coco_to_lvis_synset.json: 包含 COCO 和 LVIS 类别映射关系的 JSON 文件。

lvisv0.5+coco_train.json: 包含合并后标注的 JSON 文件。

lvis_v0.5_train.json: 包含 lvisv0.5 训练集标注信息的 JSON 文件。

lvis_v0.5_val.json: 包含 lvisv0.5 测试集标注信息的 JSON 文件。

lvis_v1_train.json: 包含 lvisv1 训练集标注信息的 JSON 文件。

lvis_v1_val.json: 包含 lvisv1 测试集标注信息的 JSON 文件。

lvis_v1_image_info_test_challenge.json: 包含可全年使用的 lvisv1 测试集标注 JSON 文件。

lvis_v1_image_info_test_dev.json: 包含仅一年一次供 LVIS Challenge 使用的 lvisv1 测试集标注 JSON 文件。

5.2.5 tao 的标注文件夹

在 data/tao/annotations 中有 9 个 JSON 文件:

test_categories.json: 包含在 TAO 测试集中会被评估的类别序列的 JSON 文件。

test_without_annotations.json: 包含测试视频的 JSON 文件。images 和 videos 域包含会在测试集中被评估的图片和视频。

test_482_classes.json: 包含测试集转换结果的 JSON 文件。

train.json: 包含 TAO 训练集中 LVIS 类别标注的 JSON 文件。

train_482_classes.json: 包含训练集转换结果的 JSON 文件。

train_with_freeform.json: 包含 TAO 训练集所有类别标注的 JSON 文件。

validation.json: 包含 TAO 验证集中 LVIS 类别标注的 JSON 文件。

validation_482_classes.json: 包含验证集转换结果的 JSON 文件。

validation_with_freeform.json: 包含 TAO 验证集所有类别标注的 JSON 文件。

5.2.6 lasot 的标注文件夹

在 data/lasot/annotations 中有 2 个 JSON 文件:

lasot_train.json: 包含 LaSOT 训练集标注信息的 JSON 文件。lasot_test.json: 包含 LaSOT 测试集标注信息的 JSON 文件。

在 data/lasot/annotations 中有 2 个 TEXT 文件:

lasot_train_infos.txt: 包含 LaSOT 训练集信息的 TEXT 文件。lasot_test_infos.txt: 包含 LaSOT 测试集信息的 TEXT 文件。

5.2.7 UAV123 的标注文件夹

在 data/UAV123/annotations 中只有 1 个 JSON 文件:

uav123.json: 包含 UAV123 数据集标注信息的 JSON 文件。

在 data/UAV123/annotations 中有 1 个 TEXT 文件:

uav123_infos.txt: 包含 UAV123 数据集信息的 TEXT 文件。

5.2.8 TrackingNet 的标注和视频帧文件夹

在 data/trackingnet/TEST/frames 文件夹下有 TrackingNet 测试集的 511 个视频目录，每个视频目录下面包含该视频所有图片。data/trackingnet/TRAIN_{*}/frames 下具有类似的文件目录结构。

在 data/trackingnet/annotations 中有 2 个 JSON 文件：

trackingnet_train.json：包含 TrackingNet 训练集标注信息的 JSON 文件。trackingnet_test.json：包含 TrackingNet 测试集标注信息的 JSON 文件。

在 data/trackingnet/annotations 中有 2 个 TEXT 文件：

trackingnet_train_infos.txt：包含 TrackingNet 训练集信息的 TEXT 文件。trackingnet_test_infos.txt：包含 TrackingNet 测试集信息的 TEXT 文件。

5.2.9 OTB100 的标注和视频帧文件夹

在 data/otb100/data 文件夹下有 OTB100 数据集的 98 个视频目录，每个视频目录下的 img 文件夹包含该视频所有图片。

在 data/otb100/data/annotations 中只有 1 个 JSON 文件：

otb100.json：包含 OTB100 数据集标注信息的 JSON 文件

在 data/otb100/annotations 中有 1 个 TEXT 文件:

otb100_infos.txt: 包含 OTB100 数据信息的 TEXT 文件。

5.2.10 GOT10k 的标注和视频帧文件夹

在 data/got10k/train 文件夹下有 GOT10k 训练集的视频目录，每个视频目录下面包含该视频所有图片。data/got10k/test 和 data/got10k/val 下具有类似的文件目录结构。

在 data/got10k/annotations 中有 3 个 JSON 文件:

got10k_train.json: 包含 GOT10k 训练集标注信息的 JSON 文件。

got10k_test.json: 包含 GOT10k 测试集标注信息的 JSON 文件。

got10k_val.json: 包含 GOT10k 验证集标注信息的 JSON 文件。

在 data/got10k/annotations 中有 5 个 TEXT 文件:

got10k_train_infos.txt: 包含 GOT10k 训练集信息的 TEXT 文件。

got10k_test_infos.txt: 包含 GOT10k 测试集信息的 TEXT 文件。

got10k_val_infos.txt: 包含 GOT10k 验证集信息的 TEXT 文件。

got10k_train_vot_infos.txt: 包含 GOT10k train_vot 划分集信息的 TEXT 文件。

got10k_val_vot_infos.txt: 包含 GOT10k val_vot 划分集信息的 TEXT 文件。

5.2.11 VOT2018 的标注和视频帧文件夹

在 data/vot2018/data 文件夹下有 VOT2018 数据集的 60 个视频目录，每个视频目录下的 color 文件夹包含该视频所有图片。

在 data/vot2018/data/annotations 中只有一个 JSON 文件:

vot2018.json: 包含 VOT2018 数据集标注信息的 JSON 文件。

在 data/vot2018/data/annotations 中只有一个 TEXT 文件:

vot2018_infos.txt: 包含 VOT2018 数据集信息的 TEXT 文件。

5.2.12 youtube_vis_2019/youtube_vis2021 的标注文件夹

在 data/youtube_vis_2019/annotations 或者 data/youtube_vis_2021/annotations 下有 3 个 JSON 文件:

youtube_vis_2019_train.json/youtube_vis_2021_train.json: 包含着 youtube_vis_2019/youtube_vis2021 训练集注释信息的 JSON 文件。

youtube_vis_2019_valid.json/youtube_vis_2021_valid.json: 包含着 youtube_vis_2019/youtube_vis2021 验证集注释信息的 JSON 文件。

youtube_vis_2019_test.json/youtube_vis_2021_test.json: 包含着 youtube_vis_2019/youtube_vis2021 测试集注释信息的 JSON 文件。

CHAPTER 6

运行现有的数据集与模型

MMTracking 为现有基准测试提供了多种算法。这些算法和基准测试在 model_zoo.md中有详细说明。以下将展示如何在现有模型和标准数据集上执行常见任务，包括：

- 使用已有模型对给定的视频或者图像文件夹进行推理。
- 在标准数据集上对已有模型进行测试（推理和评估）。
- 在标准数据集上进行训练。

6.1 推理

我们提供了对给定的视频或者包含连续图像的文件夹进行推理的演示脚本。源代码可在这里得到。

请注意，如果您使用文件夹作为输入，则该文件夹中的图像名称必须是 **可排序的**，这意味着我们可以根据文件名中包含的数字来重新排序图像。目前，我们只支持读取文件名以’.jpg’，‘.jpeg’和‘.png’结尾的图片。

6.1.1 使用 VID 模型进行推理

以下脚本可以使用视频目标检测模型对一个输入视频进行推理。

```
python demo/demo_vid.py \
    ${CONFIG_FILE}\
    --input ${INPUT} \
    --checkpoint ${CHECKPOINT_FILE} \
```

(下页继续)

(续上页)

```
    [--output ${OUTPUT}] \
    [--device ${DEVICE}] \
    [--show]
```

INPUT 和 OUTPUT 支持 mp4 视频格式和文件夹格式。

可选参数:

- OUTPUT: 可视化演示的输出路径。如果未指定 OUTPUT，使用 --show 会实时显示视频。
- DEVICE: 推理设备。可选 cpu 或者 cuda:0 等。
- --show: 是否实时显示视频。

例子:

```
python ./demo/demo_vid.py \
    ./configs/vid/selsa/selsa_faster_rcnn_r101_dc5_1x_imagenetvid.py \
    --input ${VIDEO_FILE} \
    --checkpoint ../mmtrack_output/selsa_faster_rcnn_r101_dc5_1x_imagenetvid_20201218_
↪172724-aa961bcc.pth \
    --output ${OUTPUT} \
    --show
```

6.1.2 使用 MOT/VIS 模型进行推理

以下脚本可以使用多目标跟踪模型或者视频个例分割模型对单个输入视频或者图像进行推理。

```
python demo/demo_mot_vis.py \
    ${CONFIG_FILE} \
    --input ${INPUT} \
    [--output ${OUTPUT}] \
    [--checkpoint ${CHECKPOINT_FILE}] \
    [--score-thr ${SCORE_THR} \
    [--device ${DEVICE}] \
    [--backend ${BACKEND}] \
    [--show]
```

INPUT 和 OUTPUT 支持 mp4 视频格式和文件夹格式。

可选参数:

- OUTPUT: 可视化演示的输出路径。如果未指定 OUTPUT，使用 --show 会实时显示视频。
- CHECKPOINT_FILE: 如果已经在配置文件里使用 pretrains 关键字设置了预训练模型，那么模型权重文件是可选的。
- SCORE_THR: 用于过滤跟踪框的得分阈值。

- DEVICE：推理设备。可选 cpu 或者 cuda:0 等。
- BACKEND：可视化坐标框的后端。可选 cv2 或者 plt。
- --show：是否实时显示视频。

MOT 的例子：

```
python demo/demo_mot_vis.py \
    configs/mot/deepsort/sort_faster-rcnn_fpn_4e_mot17-private.py \
    --input demo/demo.mp4 \
    --output mot.mp4 \
```

注意：当运行 demo_mot_vis.py 时，我们建议您使用包含 private 的配置文件，因为这些配置文件不需要外部的检测结果。

VIS 的例子:

假设你已经将预训练权重下载在了 checkpoints/ 文件夹下。

```
python demo/demo_mot_vis.py \
    configs/vis/masktrack_rcnn/masktrack_rcnn_r50_fpn_12e_youtubevis2019.py \
    --input ${VIDEO_FILE} \
    --checkpoint checkpoints/masktrack_rcnn_r50_fpn_12e_youtubevis2019_20211022_
↪194830-6ca6b91e.pth \
    --output ${OUTPUT} \
    --show
```

6.1.3 使用 SOT 模型进行推理

以下脚本可以使用单目标跟踪模型对单个输入视频进行推理。

```
python demo/demo_sot.py \
    ${CONFIG_FILE}\
    --input ${INPUT} \
    --checkpoint ${CHECKPOINT_FILE} \
    [--output ${OUTPUT}] \
    [--device ${DEVICE}] \
    [--show]
```

INPUT 和 OUTPUT 支持 mp4 视频格式和文件夹格式。

可选参数：

- OUTPUT：可视化演示的输出路径。如果未指定 OUTPUT，使用 --show 会实时显示视频。
- DEVICE：推理设备。可选 cpu 或者 cuda:0 等。
- --show：是否实时显示视频。

- --gt_bbox_file: 视频的真实标注文件路径。我们只使用视频的第一帧标注。如果该参数没指定，你需要手动的画出视频初始框。

例子:

```
python ./demo/demo_sot.py \
    ./configs/sot/siamese_rpn/siamese_rpn_r50_20e_lasot.py \
    --input ${VIDEO_FILE} \
    --checkpoint ../mmtrack_output/siamese_rpn_r50_1x_lasot_20211203_151612-da4b3c66.
↪pth \
    [--output ${OUTPUT}] \
    [--show] \
    [--gt_bbox_file ${GT_BBOX_FILE}]
```

6.2 测试

本节将展示如何在支持的数据集上测试现有模型。支持以下测试环境:

- 单卡 GPU
- 单节点多卡 GPU
- 多节点

在测试过程中，不同的任务共享相同的 API，我们只支持 samples_per_gpu = 1。

您可以使用以下命令来测试一个数据集。

```
# single-gpu testing
python tools/test.py ${CONFIG_FILE} [--checkpoint ${CHECKPOINT_FILE}] [--out ${RESULT_
↪FILE}] [--eval ${EVAL_METRICS}]

# multi-gpu testing
./tools/dist_test.sh ${CONFIG_FILE} ${GPU_NUM} [--checkpoint ${CHECKPOINT_FILE}] [--
↪out ${RESULT_FILE}] [--eval ${EVAL_METRICS}]
```

可选参数:

- CHECKPOINT_FILE: 模型权重文件名。在应用某些 MOT 算法时不需要定义它，而是在配置文件中指定模型权重文件名。
- RESULT_FILE: pickle 格式的输出结果的文件名，如果不专门指定，结果将不会被专门保存成文件。
- EVAL_METRICS: 用于评估结果的指标。允许的值取决于数据集，例如, bbox 适用于 ImageNet VID, track 适用于 LaSOT, bbox and track 都适用于 MOT17。
- --cfg-options: 如果指定，可选配置的键值对将被合并进配置文件中。

- `--eval-options`：如果指定，可选评估配置的键值对将作为 dataset.evaluate() 函数的参数，此参数只适用于评估。
- `--format-only`：如果指定，结果将被格式化为官方格式。

6.2.1 测试 VID 模型示例

假设您已经下载模型权重文件至文件夹 `checkpoints/` 里。

1. 在 ImageNet VID 上测试 DFF，并且评估 bbox mAP。

```
python tools/test.py configs/vid/dff/dff_faster_rcnn_r101_dc5_1x_imagenetvid.py \
    --checkpoint checkpoints/dff_faster_rcnn_r101_dc5_1x_imagenetvid_20201218_
↪172720-ad732e17.pth \
    --out results.pkl \
    --eval bbox
```

2. 使用 8 GPUs 测试 DFF，并且评估 bbox mAP。

```
./tools/dist_test.sh configs/vid/dff/dff_faster_rcnn_r101_dc5_1x_imagenetvid.py 8␣
↪\
    --checkpoint checkpoints/dff_faster_rcnn_r101_dc5_1x_imagenetvid_20201218_
↪172720-ad732e17.pth \
    --out results.pkl \
    --eval bbox
```

6.2.2 测试 MOT 模型示例

1. 在 MOT17 上测试 Tracktor，并且评估 CLEAR MOT 指标。

```
python tools/test.py configs/mot/tracktor/tracktor_faster-rcnn_r50_fpn_4e_mot17-
↪public-half.py \
    --eval track
```

2. 使用 8 GPUs 测试 Tracktor，并且评估 CLEAR MOT 指标。

```
./tools/dist_test.sh  \
    configs/mot/tracktor/tracktor_faster-rcnn_r50_fpn_4e_mot17-public-half.py 8 \
    --eval track
```

3. 如果想使用自定义的检测器和重识别模型来测试 Trackor，你需要在 config 中更改相应的（关键字-值）对，如下所示：

```
model = dict(
    detector=dict(
        init_cfg=dict(
            type='Pretrained',
            checkpoint='/path/to/detector_model')),
    reid=dict(
        init_cfg=dict(
            type='Pretrained',
            checkpoint='/path/to/reid_model'))
    )
```

6.2.3 测试 SOT 模型示例

1. 在 LaSOT 上测试 SiameseRPN++，并且评估 success 和 normed precision。

```
python tools/test.py configs/sot/siamese_rpn/siamese_rpn_r50_20e_lasot.py \
    --checkpoint checkpoints/siamese_rpn_r50_1x_lasot_20211203_151612-da4b3c66.
↪pth \
    --out results.pkl \
    --eval track
```

2. 使用 8 GPUs 测试 SiameseRPN++，并且评估 success 和 normed precision。

```
./tools/dist_test.sh configs/sot/siamese_rpn/siamese_rpn_r50_20e_lasot.py 8 \
    --checkpoint checkpoints/siamese_rpn_r50_1x_lasot_20211203_151612-da4b3c66.
↪pth \
    --out results.pkl \
    --eval track
```

6.2.4 测试 VIS 模型示例

假设你已经将预训练权重下载在了 `checkpoints/` 文件夹下。

1. 在 YouTube-VIS 2019 上测试 MaskTrack R-CNN，并且生成一个用于提交结果的 zip 文件。

```
python tools/test.py \
    configs/vis/masktrack_rcnn/masktrack_rcnn_r50_fpn_12e_youtubevis2019.py \
    --checkpoint checkpoints/masktrack_rcnn_r50_fpn_12e_youtubevis2019_20211022_
↪194830-6ca6b91e.pth \
    --out ${RESULTS_PATH}/results.pkl \
    --format-only \
    --eval-options resfile_path=${RESULTS_PATH}
```

2. 使用 8 GPUs 在 YouTube-VIS 2019 上测试 MaskTrack R-CNN，并且生成一个用于提交结果的 zip 文件。

```
./tools/dist_test.sh \
    configs/vis/masktrack_rcnn/masktrack_rcnn_r50_fpn_12e_youtubevis2019.py \
    --checkpoint checkpoints/masktrack_rcnn_r50_fpn_12e_youtubevis2019_20211022_
↪194830-6ca6b91e.pth \
    --out ${RESULTS_PATH}/results.pkl \
    --format-only \
    --eval-options resfile_path=${RESULTS_PATH}
```

6.3 训练

MMTracking 也为训练模型提供了开箱即用的工具。本节将展示如何在标准数据集（即 MOT17）上训练 预定义模型（在 configs 下）。

默认情况下，我们在每个 epoch 之后在验证集上评估模型，您可以通过在训练配置文件中添加 interval 参数来更改评估间隔。

```
evaluation = dict(interval=12)  # 每 12 个 epoch 评估一次模型。
```

重要提醒：配置文件中的默认学习率是针对使用 8 个 GPU 的。根据 Linear Scaling Rule，如果您使用不同数量的 GPU 或每个 GPU 使用不同数量的图片，则需要设置与 batch size 成正比的学习率，例如：`lr=0.01` 用于 8 个 GPU * 1 img/gpu，`lr=0.04` 用于 16 个 GPU * 2 imgs/gpu。

6.3.1 单 GPU 训练

```
python tools/train.py ${CONFIG_FILE} [optional arguments]
```

在训练期间，日志文件和模型权重文件将保存到工作目录中，该目录由配置文件中的 `work_dir` 或通过 CLI 的 `--work-dir` 参数指定。

6.3.2 多 GPU 训练

我们提供了 `tools/dist_train.sh` 来在多个 GPU 上启动训练。基本用法如下：

```
bash ./tools/dist_train.sh \
    ${CONFIG_FILE} \
    ${GPU_NUM} \
    [optional arguments]
```

可选参数与上述相同。

如果您想在一台机器上启动多个任务，例如，在有 8 个 GPU 的机器上进行 2 个 4-GPU 训练的任务，您需要为每个任务指定不同的端口（默认为 29500）以避免通信冲突。

如果您使用 dist_train.sh 来启动训练任务，您可以使用以下命令来设置端口：

```
CUDA_VISIBLE_DEVICES=0,1,2,3 PORT=29500 ./tools/dist_train.sh ${CONFIG_FILE} 4
CUDA_VISIBLE_DEVICES=4,5,6,7 PORT=29501 ./tools/dist_train.sh ${CONFIG_FILE} 4
```

6.3.3 多节点训练

如果您想使用由 ethernet 连接起来的多台机器，您可以使用以下命令:

在第一台机器上:

```
NNODES=2 NODE_RANK=0 PORT=$MASTER_PORT MASTER_ADDR=$MASTER_ADDR bash tools/dist_train.
↪sh $CONFIG $GPUS
```

在第二台机器上:

```
NNODES=2 NODE_RANK=1 PORT=$MASTER_PORT MASTER_ADDR=$MASTER_ADDR bash tools/dist_train.
↪sh $CONFIG $GPUS
```

但是，如果您不使用高速网路连接这几台机器的话，训练将会非常慢。

如果您使用的是 slurm 来管理多台机器，您可以使用同在单台机器上一样的命令来启动任务，但是您必须得设置合适的环境变量和参数，具体可以参考slurm_train.sh。

6.3.4 使用 Slurm 管理任务

Slurm 是一款优秀的计算集群任务调度系统。在 Slurm 管理的集群上，您可以使用 slurm_train.sh 来启动训练任务，并且它同时支持单节点和多节点训练。

基本用法如下：

```
[GPUS=${GPUS}] ./tools/slurm_train.sh ${PARTITION} ${JOB_NAME} ${CONFIG_FILE} ${WORK_
↪DIR}
```

您可以查看源代码以了解全部的参数和环境变量。

使用 Slurm 时，需要通过以下方式之一设置端口选项：

1. 通过 --options 设置端口。推荐使用此方式，因为它不会更改原始配置。

```
CUDA_VISIBLE_DEVICES=0,1,2,3 GPUS=4 ./tools/slurm_train.sh ${PARTITION} ${JOB_
↪NAME} config1.py ${WORK_DIR} --options 'dist_params.port=29500'
CUDA_VISIBLE_DEVICES=4,5,6,7 GPUS=4 ./tools/slurm_train.sh ${PARTITION} ${JOB_
↪NAME} config2.py ${WORK_DIR} --options 'dist_params.port=29501'
```

2. 修改配置文件以设置不同的通信端口。

   在 config1.py 中设置

```
dist_params = dict(backend='nccl', port=29500)
```

   在 config2.py 中设置

```
dist_params = dict(backend='nccl', port=29501)
```

   然后可以使用 config1.py 和 config2.py 启动两个任务。

```
CUDA_VISIBLE_DEVICES=0,1,2,3 GPUS=4 ./tools/slurm_train.sh ${PARTITION} ${JOB_
↪NAME} config1.py ${WORK_DIR}
CUDA_VISIBLE_DEVICES=4,5,6,7 GPUS=4 ./tools/slurm_train.sh ${PARTITION} ${JOB_
↪NAME} config2.py ${WORK_DIR}
```

6.3.5 训练 VID 模型示例

1. 在 ImageNet VID 和 ImageNet DET 上训练 DFF，接着在最后一个 epoch 评估 bbox mAP.

```
bash ./tools/dist_train.sh ./configs/vid/dff/dff_faster_rcnn_r101_dc5_1x_imagenetvid.
↪py 8 --work-dir ./work_dirs/
```

6.3.6 训练 MOT 模型示例

对于像 MOT、SORT、DeepSORT 以及 Trackor 这样的 MOT 方法，你需要训练一个检测器和一个 reid 模型，而非直接训练 MOT 模型。

1. 训练检测器

   如果你想要为多目标跟踪器训练检测器，为了兼容 MMDetection, 你只需要在 config 里面增加一行代码 USE_MMDET=True, 然后使用与 MMDetection 相同的方式运行它。可参考示例 faster_rcnn_r50_fpn.py。

   请注意 MMTracking 和 MMDetection 在 base config 上有些许不同：detector 仅仅是 model 的一个子模块。例如，MMDetection 中的 Faster R-CNN 的 config 如下：

```
model = dict(
    type='FasterRCNN',
    ...
)
```

   但在 MMTracking 中，config 如下：

```
model = dict(
    detector=dict(
        type='FasterRCNN',
        ...
    )
)
```

这里有一个在 MOT17 上训练检测器模型，并在每个 epoch 结束后评估 bbox mAP 的范例：

```
bash ./tools/dist_train.sh ./configs/det/faster-rcnn_r50_fpn_4e_mot17-half.py 8 \
    --work-dir ./work_dirs/
```

2. 训练 ReID 模型

   你可能需要在 MOT 或其它实际应用中训练 ReID 模型。我们在 MMTracking 中也支持 ReID 模型的训练，这是基于 MMClassification 实现的。

   这里有一个在 MOT17 上训练检测器模型，并在每个 epoch 结束后评估 bbox mAP 的范例：

```
bash ./tools/dist_train.sh ./configs/reid/resnet50_b32x8_MOT17.py 8 \
    --work-dir ./work_dirs/
```

3. 完成检测器和 ReID 模型训练后，可参考测试 MOT 模型示例来测试多目标跟踪器。

6.3.7 训练 SOT 模型示例

1. 在 COCO、ImageNet VID 和 ImageNet DET 上训练 SiameseRPN++，然后从第 10 个 epoch 到第 20 个 epoch 评估其 success、precision 和 normed precision。

```
bash ./tools/dist_train.sh ./configs/sot/siamese_rpn/siamese_rpn_r50_20e_lasot.py␣
↪8 \
    --work-dir ./work_dirs/
```

6.3.8 训练 VIS 模型示例

1. 在 YouTube-VIS 2019 数据集上训练 MaskTrack R-CNN。由于 YouTube-VIS 没有提供 validation 集的注释文件，因此在训练过程中不会进行评估。

```
bash ./tools/dist_train.sh ./configs/vis/masktrack_rcnn/masktrack_rcnn_r50_fpn_
↪12e_youtubevis2019.py 8 \
    --work-dir ./work_dirs/
```

CHAPTER 7

使用自定义数据集和模型运行

在本节中，您将了解如何使用自定义数据集和模型进行推理、测试和训练：

基本步骤如下：

1. 准备自定义数据集（如果适用）
2. 准备自定义模型（如果适用）
3. 准备配置文件
4. 训练新模型
5. 测试、推理新模型

7.1 1. 准备自定义数据集

在 MMTracking 中支持新数据集的方式有以下两种：

1. 将数据集重组为 CocoVID 格式。
2. 实现一个新的数据集。

通常我们建议使用第一种方法，它比第二种方法容易实现。

tutorials/customize_dataset.md 中提供了有关自定义数据集的详细教程。

7.2 2. 准备自定义模型

我们提供了不同任务下自定义模型的教程：

- tutorials/customize_mot_model.md
- tutorials/customize_sot_model.md
- tutorials/customize_vid_model.md

7.3 3. 准备配置文件

下一步是准备配置文件，从而可以成功加载数据集或模型。tutorials/config.md 提供了有关配置系统的更多详细教程。

7.4 4. 训练新模型

要使用新的配置文件训练模型，您只需运行

```
python tools/train.py ${NEW_CONFIG_FILE}
```

更详细的用法请参考前面的训练说明。

7.5 5. 测试和推理

要测试经过训练的模型，您只需运行

```
python tools/test.py ${NEW_CONFIG_FILE} ${TRAINED_MODEL} --eval bbox track
```

更详细的用法请参考前面的测试或推理说明。

CHAPTER 8

了解配置文件

我们使用 python 文件作为我们的配置系统。你可以在 $MMTracking/configs 下找到所有配置文件。

我们将模块化和继承融入我们的配置系统，这将方便我们进行各种实验。你可以运行 `python tools/print_config.py /PATH/TO/CONFIG` 来查看完整的配置。

8.1 通过脚本参数修改配置

使用“tools/train.py”或“tools/test.py”提交任务时，你可以指定 `--cfg-options` 来原地修改配置。

- 更新配置文件中字典的键值。

  可以按照原始配置中字典键的顺序修改配置选项。例如 `--cfg-options model.detector.backbone.norm_eval=False` 将模型骨干网络中的 BN 模块更改为训练模式。

- 更新配置文件列表中的字典键值。

  在配置文件中，一些配置字典被组成一个列表。例如，测试时的流水线 `data.test.pipeline` 通常是一个列表例如 `[dict(type='LoadImageFromFile'), ...]` 。如果你想把流水线中的 `'LoadImageFromFile'` 更改为 `'LoadImageFromWebcam'`，你可以使用 `--cfg-options data.test.pipeline.0.type=LoadImageFromWebcam`。

- 更新列表/元组的值。

  如果要更新的值是列表或元组。例如配置文件中通常设置 `workflow=[('train', 1)]` 。如果你想改变这个键值，你可以指定 `--cfg-options workflow="[(train,1),(val,1)]"` 。注意引号”是为了支持列表/元组数据类型，并且在指定键值的引号内 **不允许**有空格。

8.2 配置文件结构

config/_base_ 下有 3 种基本组件类型分别是 dataset，model，default_runtime。许多方法可以通过这些文件构筑，例如 DFF、FGFA、SELSA、SORT、DeepSORT。由来自 _base_ 的组件组成的配置称为 *primitive* 。

对于同一文件夹下的所有配置，建议只有一个 *primitive* 配置。所有其他配置都应该从 *primitive* 配置中继承。在这种方式下，最大的继承级别是 3。

为了便于理解，我们建议开发者继承现有方法。例如，如果基于 Faster R-CNN 做了一些修改，用户可以先通过指定 _base_ = ../../_base_/models/faster_rcnn_r50_dc5.py 来继承基本的 Faster R-CNN 结构，然后在配置文件中修改必要的字段。

如果您正在构建一个与任何现有方法都不共享结构的新方法，你可以在 configs 下创建一个文件夹 xxx_rcnn，

详细文档请参考mmcv。

8.3 配置文件名样式

我们遵循以下样式来命名配置文件。建议开发者遵循相同的样式。

```
{model}_[model setting]_{backbone}_{neck}_[norm setting]_[misc]_[gpu x batch_per_gpu]_
↪{schedule}_{dataset}
```

{xxx} 是必填字段，[yyy] 是可选字段。

- {model}：模型类型，例如 dff、tracktor、siamese_rpn 等。
- [model setting]：某些模型中的特定设置，例如 dff、tracktor 中使用的 faster_rcnn。
- {backbone}：主干网络，例如 r50 (ResNet-50)、x101 (ResNeXt-101)。
- {neck}：模型颈部，例如 fpn、c5。
- [norm_setting]：标准化设置，除了 bn（Batch Normalization）不需要注明以外，其他标准化类型比如 gn（Group Normalization），syncbn（Synchronized Batch Normalization）都应注明。gn-head/gn-neck 表示 GN 仅应用于模型头部/模型颈部，而 gn-all 表示 GN 应用于整个模型，例如主干网络，模型颈部，模型头部。
- [misc]：模型的其他设置/插件，例如 dconv、gcb、attention、albu、mstrain。
- [gpu x batch_per_gpu]：GPU 数目以及每个 GPU 的样本数，默认使用 8x2。
- {schedule}：训练时长，选项为 4e、7e、20e 等。20e 表示 20 个周期。
- {dataset}：数据集如 imagenetvid、mot17、lasot。

8.4 配置文件详细解析

不同任务的配置文件结构请参考相应页面。

视频目标检测

多目标跟踪

单目标跟踪

8.5 常见问题

8.5.1 忽略基础配置中的一些字段

你可以设置 `_delete_=True` 来忽略基本配置中的某些字段。详细内容请参考 mmcv

8.5.2 使用配置中的中间变量

配置文件中使用了一些中间变量，例如数据集中的 `train_pipeline/test_pipeline`。值得注意的是，在子配置中修改中间变量时，用户需要再次将中间变量传递到相应的字段中。例如，我们想使用自适应步长的测试策略来测试 SELSA。ref_img_sampler 是我们想要修改的中间变量。

```
_base_ = ['./selsa_faster_rcnn_r50_dc5_1x_imagenetvid.py']

# dataset settings
ref_img_sampler = dict(
    _delete_=True,
    num_ref_imgs=14,
    frame_range=[-7, 7],
    method='test_with_adaptive_stride')
data = dict(
    val=dict(
        ref_img_sampler=ref_img_sampler),
    test=dict(
        ref_img_sampler=ref_img_sampler))
```

我们首先需要定义新的 `ref_img_sampler` 然后将其赋值到 `data` 字段下对应位置。

CHAPTER 9

自定义数据集

对于自定义的数据集，你既可以将其转换成 CocoVID 格式，也可以实现一个全新的数据集。对于 MMTracking，我们建议将数据离线转换成 CocoVID 格式，这样就可以直接使用 `CocoVideoDataset` 类了。在这种情况下，你只需要修改配置文件中的数据标注路径和 `classes` 类别即可。

9.1 将数据集转换为 CocoVID 样式

9.1.1 CocoVID 标注文件

CocoVID 风格的标注文件需要以下键：

- `videos`：视频序列。每个视频都是包含 `name`、`id` 键的一个字典，键为 `name`、`id`。可选键有 `fps`、`width` 和 `height`。
- `images`：图像序列。每张图像都是包含 `file_name`、`height`、`width`、`id`、`frame_id` 和 `video_id` 键的一个字典。请注意，`frame_id` 的**索引是从 0 开始的**。
- `annotations`：实例标注序列。每个标注都是包含 `bbox`、`area`、`id`、`category_id`、`instance_id`、`image_id` 和 `video_id` 键的一个字典。其中 `instance_id` 仅用于跟踪任务。
- `categories`：类别序列。每个类别都是包含 `id`、`name` 键的一个字典。

此处 提供了一个简单实例。

此处 提供了转换现有数据集的实例。

9.1.2 修改配置

数据预处理后，用户需要进一步修改配置文件才能使用自定义数据集。这里我们展示了一个使用 5 个类的自定义数据集的实例，假设其以及被转换成 CocoVID 格式。

在 `configs/my_custom_config.py`:

```
...
# dataset settings
dataset_type = 'CocoVideoDataset'
classes = ('a', 'b', 'c', 'd', 'e')
...
data = dict(
    samples_per_gpu=2,
    workers_per_gpu=2,
    train=dict(
        type=dataset_type,
        classes=classes,
        ann_file='path/to/your/train/data',
        ...),
    val=dict(
        type=dataset_type,
        classes=classes,
        ann_file='path/to/your/val/data',
        ...),
    test=dict(
        type=dataset_type,
        classes=classes,
        ann_file='path/to/your/test/data',
        ...))
...
```

9.2 使用数据集包装器

MMTracking 还支持一些数据集包装器来混合数据集或修改数据集分布。目前支持三个数据集包装器，如下所示：

- `RepeatDataset`：简单地重复整个数据集。
- `ClassBalancedDataset`：以类平衡的方式重复数据集。
- `ConcatDataset`：拼接多个数据集。

9.2.1 重复数据集

我们使用 RepeatDataset 作为包装器来重复数据集。例如，假设原始数据集是 Dataset_A，我们需要重复该数据集，可以将配置做如下修改：

```
dataset_A_train = dict(
        type='RepeatDataset',
        times=N,
        dataset=dict(  # This is the original config of Dataset_A
            type='Dataset_A',
            ...
            pipeline=train_pipeline
        )
    )
```

9.2.2 类平衡数据集

我们使用 ClassBalancedDataset 作为包装器来实现类平衡数据集。要重复的数据集需要实例化函数 self.get_cat_ids(idx) 来支持 ClassBalancedDataset 类。例如，要使用配置 oversample_thr=1e-3 重复数据集 Dataset_A，配置如下所示：

```
dataset_A_train = dict(
        type='ClassBalancedDataset',
        oversample_thr=1e-3,
        dataset=dict(  # This is the original config of Dataset_A
            type='Dataset_A',
            ...
            pipeline=train_pipeline
        )
    )
```

9.2.3 拼接数据集

有三种方法可以拼接数据集。

1. 如果要拼接的数据集类型相同，注释文件不同，可以将配置文件按照如下方式修改：

```
dataset_A_train = dict(
    type='Dataset_A',
    ann_file = ['anno_file_1', 'anno_file_2'],
    pipeline=train_pipeline
)
```

有两种方式支持测试或评估拼接后的数据集，默认的方式为对每个数据集进行单独评估。如果你想要将拼接后的数据集作为一个整体进行评测，你可以设置 `separate_eval=False` 具体修改方式如下：

```
dataset_A_train = dict(
    type='Dataset_A',
    ann_file = ['anno_file_1', 'anno_file_2'],
    separate_eval=False,
    pipeline=train_pipeline
)
```

2. 如果要连接的数据集类型不同，可以将配置文件按照如下方式修改：

```
dataset_A_train = dict()
dataset_B_train = dict()

data = dict(
    imgs_per_gpu=2,
    workers_per_gpu=2,
    train = [
        dataset_A_train,
        dataset_B_train
    ],
    val = dataset_A_val,
    test = dataset_A_test
    )
```

通过该方法拼接的数据集同样支持两种方式评测，默认的方式为对每个数据集进行单独评估。

3. 我们也支持显式定义 `ConcatDataset` 类。

```
dataset_A_val = dict()
dataset_B_val = dict()

data = dict(
    imgs_per_gpu=2,
    workers_per_gpu=2,
    train=dataset_A_train,
    val=dict(
        type='ConcatDataset',
        datasets=[dataset_A_val, dataset_B_val],
        separate_eval=False))
```

这种方式允许用户通过设置 `separate_eval=False` 来将所有数据集进行统一的评测。

请注意：

1. `separate_eval=False` 假设了数据集在评测使用 `self.data_infos` 方法。由于 `CocoVID` 数据集

不完全依赖于 `self.data_infos` 方法进行评测。因此，`CocoVID` 数据集不支持将数据集统一起来评测。同时，我们也不建议将不同类型的数据集并将它们作为一个整体进行评测。

2. 由于 `ClassBalancedDataset` 和 `RepeatDataset` 不支持评测，因此由这些数据集拼接而成的数据集也不支持评测。

这里有一个更复杂的例子：分别将 `Dataset_A` 和 `Dataset_B` 重复 N 次和 M 次，然后拼接重复的数据集，相关配置如下所示：

```
dataset_A_train = dict(
    type='RepeatDataset',
    times=N,
    dataset=dict(
        type='Dataset_A',
        ...
        pipeline=train_pipeline
    )
)
dataset_A_val = dict(
    ...
    pipeline=test_pipeline
)
dataset_A_test = dict(
    ...
    pipeline=test_pipeline
)
dataset_B_train = dict(
    type='RepeatDataset',
    times=M,
    dataset=dict(
        type='Dataset_B',
        ...
        pipeline=train_pipeline
    )
)
data = dict(
    imgs_per_gpu=2,
    workers_per_gpu=2,
    train = [
        dataset_A_train,
        dataset_B_train
    ],
    val = dataset_A_val,
    test = dataset_A_test
)
```

9.3 现有数据集的子集

对于现有的数据集，我们可以修改其配置文件中目标种类来训练子数据集。例如，如果你只想训练当前数据集的三个类，你可以将当前数据集配置文件中的目标中类做相应的修改。数据集会自动过滤掉其他类的真实标签框。

```
classes = ('person', 'bicycle', 'car')
data = dict(
    train=dict(classes=classes),
    val=dict(classes=classes),
    test=dict(classes=classes))
```

MMTracking 还支持从文件中读取类，这在实际应用中很常见。例如，假设 `classes.txt` 包含如下类名：

```
person
bicycle
car
```

用户可以将类设置为包含类的文件路径，数据集将加载并自动将其转换为列表。

```
classes = 'path/to/classes.txt'
data = dict(
    train=dict(classes=classes),
    val=dict(classes=classes),
    test=dict(classes=classes))
```

CHAPTER 10

自定义数据预处理流程

MMTracking 中有两种数据流水线：

- 单张图片，这与 MMDetection 中的大部分情况相同
- 成对/多张图片

10.1 单张图片的数据预处理流程

对于单张图片，可以参考MMDetection 教程。此外，MMTracking 还有些许不同之处：

- 我们的 `VideoCollect` 实现方法和 MMdetection 中的 `Collect` 相似，但是更适用于视频感知任务。例如：`frame_id` 和 `is_video_data` 属性会被默认收集出来。

10.2 多张图片的数据预处理流程

在多数情况下，我们需要同时处理多张图片。这主要是因为我们需要在同一视频中针对关键帧采样多个参考帧，以便于后续训练和推理。请先察看单张图片的预处理实现，因为多张图片的预处理实现也是基于此。我们接下来将详细介绍整体流程。

10.2.1 1. 采样参考帧

一旦我们得到关键帧的标注，我们将采样和加载参考帧的标注。

以 CocoVideoDataset 为例，我们用函数 ref_img_sampling 来采样和加载参考图片的标注。

```
from mmdet.datasets import CocoDataset

class CocoVideoDataset(CocoDataset):

    def __init__(self,
                 ref_img_sampler=None,
                 *args,
                 **kwargs):
        super().__init__(*args, **kwargs)
        self.ref_img_sampler = ref_img_sampler


    def ref_img_sampling(self, **kwargs):
        pass

    def prepare_data(self, idx):
        img_info = self.data_infos[idx]
        if self.ref_img_sampler is not None:
            img_infos = self.ref_img_sampling(img_info, **self.ref_img_sampler)
        ...
```

在这种情况下，加载的标注不再是一个 dict, 而是一个包含关键图片和参考图片的 list[dict]。这个列表的首个元素就是关键图片的标注。

10.2.2 2. 序列处理和收集数据

在这一步，我们执行图片转换并且收集图片信息。

单张图片预处理流程是接收字典作为输入，并输出字典，然后进入后续的图片转换; 与之不同，序列的预处理流程是接收一个包含字典的列表作为输入，并输出一个包含字典的列表，然后进入后续的图片转换。

序列的预处理流程通常继承于 MMDetection, 但是会对列表元素做循环处理。

```
from mmdet.datasets.builder import PIPELINES
from mmdet.datasets.pipelines import LoadImageFromFile

@PIPELINES.register_module()
class LoadMultiImagesFromFile(LoadImageFromFile):

```

(下页继续)

(续上页)

```
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

    def __call__(self, results):
        outs = []
        for _results in results:
            _results = super().__call__(_results)
            outs.append(_results)
        return outs
```

有时，你需要增加参数 share_params 来决定是否共享图片转换的随机种子。

10.2.3 3. 拼接参考图片（如果需要）

如果参考图片超过一个，我们利用 ConcatVideoReferences 来以字典形式收集所有参考图片。经处理后，该包含关键图片和参考图片的列表总长度为 2。

10.2.4 4. 将输出结果格式化成一个字典

最后，我们利用 SeqDefaultFormatBundle 来将数据的列表形式转换为字典形式，作为后续模型的输入。这里有一个数据全部处理流水线的示例：

```
train_pipeline = [
    dict(type='LoadMultiImagesFromFile'),
    dict(type='SeqLoadAnnotations', with_bbox=True, with_track=True),
    dict(type='SeqResize', img_scale=(1000, 600), keep_ratio=True),
    dict(type='SeqRandomFlip', share_params=True, flip_ratio=0.5),
    dict(type='SeqNormalize', **img_norm_cfg),
    dict(type='SeqPad', size_divisor=16),
    dict(
        type='VideoCollect',
        keys=['img', 'gt_bboxes', 'gt_labels', 'gt_instance_ids']),
    dict(type='ConcatVideoReferences'),
    dict(type='SeqDefaultFormatBundle', ref_prefix='ref')
]
```

CHAPTER 11

自定义视频目标检测模型

我们通常将模型组件分为 3 类：

- 检测器：通常是从一张图片中检出物体的检测组件，例如：Faster R-CNN。
- 运动估计器：计算两张图片之间的运动信息的组件，例如：FlowNetSimple。
- 聚合器：聚合多张图片特征的组件，例如：EmbedAggregator。

11.1 增加一个新的检测器

请参考MMDetection 教程来开发新检测器

11.2 增加一个新的运动估计器

11.2.1 1. 定义一个运动估计模型（例如：MyFlowNet）

新建一个文件 `mmtrack/models/motion/my_flownet.py`。

```
from mmcv.runner import BaseModule

from ..builder import MOTION

@MOTION.register_module()
```

(下页继续)

(续上页)

```
class MyFlowNet(BaseModule):

    def __init__(self,
                 arg1,
                 arg2):
        pass

    def forward(self, inputs):
        # implementation is ignored
        pass
```

11.2.2 2. 引入模块

你可以在 mmtrack/models/motion/__init__.py 中加入下面一行。

```
from .my_flownet import MyFlowNet
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.models.motion.my_flownet.py'],
    allow_failed_imports=False)
```

11.2.3 3. 更改原始 config 文件

```
motion=dict(
    type='MyFlowNet',
    arg1=xxx,
    arg2=xxx)
```

11.3 增加一个新的聚合器

11.3.1 1. 定义一个聚合器

创建一个新文件 mmtrack/models/aggregators/my_aggregator.py。

```
from mmcv.runner import BaseModule

from ..builder import AGGREGATORS
```

(下页继续)

(续上页)

```
@AGGREGATORS.register_module()
class MyAggregator(BaseModule):

    def __init__(self,
                 arg1,
                 arg2):
        pass

    def forward(self, inputs):
        # implementation is ignored
        pass
```

11.3.2 2. 引入模块

你可以在 mmtrack/models/aggregators/__init__.py 中加入下面一行。

```
from .my_aggregator import MyAggregator
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.models.aggregators.my_aggregator.py'],
    allow_failed_imports=False)
```

11.3.3 3. 更改原始 config 文件

```
aggregator=dict(
    type='MyAggregator',
    arg1=xxx,
    arg2=xxx)
```

CHAPTER 12

自定义多目标跟踪模型

我们通常将模型组件分为 5 类：

- 跟踪器：利用以下组件提取出来的线索来关联视频帧间目标的组件。
- 检测器：通常是从一张图片中检出物体的检测器，例如：Faster R-CNN。
- 运动估计器：计算相邻帧运动信息的组件，例如：卡尔曼滤波器。
- 重识别器：从裁剪图片中抽取特征的的独立重识别模型，例如：BaseReID。
- 跟踪头：用于抽取跟踪线索但是和检测器共享骨干网络的组件。例如：一个特征分支头或者回归分支头。

12.1 增加一个新的跟踪器

12.1.1 1. 定义一个跟踪器

创建一个新文件 `mmtrack/models/mot/trackers/my_tracker.py`。`BaseTracker` 是提供跨视频跟踪基础 APIs，我们推荐新的跟踪器继承该类，用户可以参考 BaseTracker 的文档来了解细节。

```
from mmtrack.models import TRACKERS
from .base_tracker import BaseTracker

@TRACKERS.register_module()
class MyTracker(BaseTracker):
```

(下页继续)

(续上页)

```
    def __init__(self,
                 arg1,
                 arg2,
                 *args,
                 **kwargs):
        super().__init__(*args, **kwargs)
        pass

    def track(self, inputs):
        # implementation is ignored
        pass
```

12.1.2 2. 引入模块

你可以在 mmtrack/models/mot/trackers/__init__.py 中加入下面一行。

```
from .my_tracker import MyTracker
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.models.mot.trackers.my_tracker.py'],
    allow_failed_imports=False)
```

12.1.3 3. 更改原始 config 文件

```
tracker=dict(
    type='MyTracker',
    arg1=xxx,
    arg2=xxx)
```

12.2 增加一个新的检测器

请参考MMDetection 教程来开发新检测器

12.3 增加一个新的运动估计器

12.3.1 1. 定义一个运动估计模型（例如：MyFlowNet）

新建一个文件 mmtrack/models/motion/my_flownet.py。

如果该运动估计模型是一个深度学习模块，你可以继承 mmcv.runner 的 BaseModule，否则继承 Object。

```
from mmcv.runner import BaseModule

from ..builder import MOTION

@MOTION.register_module()
class MyFlowNet(BaseModule):

    def __init__(self,
                 arg1,
                 arg2):
        pass

    def forward(self, inputs):
        # implementation is ignored
        pass
```

12.3.2 2. 引入模块

你可以在 mmtrack/models/motion/__init__.py 中加入下面一行。

```
from .my_flownet import MyFlowNet
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.models.motion.my_flownet.py'],
    allow_failed_imports=False)
```

12.3.3 3. 更改原始 config 文件

```
motion=dict(
    type='MyFlowNet',
    arg1=xxx,
    arg2=xxx)
```

12.4 增加一个新的重识别模型

12.4.1 1. 定义一个识别模型（例如：MyReID）

新建一个文件 mmtrack/models/motion/my_flownet.py。

```
from mmcv.runner import BaseModule

from ..builder import REID

@REID.register_module()
class MyReID(BaseModule):

    def __init__(self,
                 arg1,
                 arg2):
        pass

    def forward(self, inputs):
        # implementation is ignored
        pass
```

12.4.2 2. 引入模块

你可以在 mmtrack/models/reid/__init__.py 中加入下面一行。

```
from .my_reid import MyReID
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.models.reid.my_reid.py'],
    allow_failed_imports=False)
```

12.4.3 3. 更改原始 config 文件

```
motion=dict(
    type='MyReID',
    arg1=xxx,
    arg2=xxx)
```

12.5 增加一个新的跟踪头

12.5.1 1. 定义一个跟踪头（例如：MyHead）

新建一个文件 mmtrack/models/track_heads/my_head.py。

```
from mmcv.runner import BaseModule

from mmdet.models import HEADS

@HEADS.register_module()
class MyHead(BaseModule):

    def __init__(self,
                 arg1,
                 arg2):
        pass

    def forward(self, inputs):
        # implementation is ignored
        pass
```

12.5.2 2. 引入模块

你可以在 mmtrack/models/track_heads/__init__.py 中加入下面一行。

```
from .my_head import MyHead
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.models.track_heads.my_head.py'],
    allow_failed_imports=False)
```

12.5.3 3. 更改原始 config 文件

```
motion=dict(
    type='MyHead',
    arg1=xxx,
    arg2=xxx)
```

12.6 增加一个新的损失函数

12.6.1 1. 定义一个损失函数

假定你想要增加一个新的损失函数 MyLoss 来进行边界框回归。为此，你需要定义一个文件 mmtrack/models/losses/my_loss.py。装饰器 weighted_loss 可以对损失函数输出结果做基于单个元素的加权平均。

```
import torch
import torch.nn as nn

from mmdet.models import LOSSES, weighted_loss

@weighted_loss
def my_loss(pred, target):
    assert pred.size() == target.size() and target.numel() > 0
    loss = torch.abs(pred - target)
    return loss

@LOSSES.register_module()
class MyLoss(nn.Module):

    def __init__(self, reduction='mean', loss_weight=1.0):
        super(MyLoss, self).__init__()
        self.reduction = reduction
        self.loss_weight = loss_weight

    def forward(self,
                pred,
                target,
                weight=None,
                avg_factor=None,
                reduction_override=None):
        assert reduction_override in (None, 'none', 'mean', 'sum')
        reduction = (
```

(下页继续)

(续上页)

```
            reduction_override if reduction_override else self.reduction)
        loss_bbox = self.loss_weight * my_loss(
            pred, target, weight, reduction=reduction, avg_factor=avg_factor)
        return loss_bbox
```

12.6.2 2. 引入模块

你可以在 mmtrack/models/losses/__init__.py 中加入下面一行。

```
from .my_loss import MyLoss, my_loss
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports=dict(
    imports=['mmtrack.models.losses.my_loss'],
    allow_failed_imports=False)
```

12.6.3 3. 更改原始 config 文件

为了使用新的损失函数，你需要更改 loss_xxx 区域。假设 MyLoss 是用于回归任务，则需在 head 区域中更改 loss_bbox。

```
loss_bbox=dict(type='MyLoss', loss_weight=1.0))
```

CHAPTER 13

自定义单目标跟踪模型

我们通常将模型组件分为 4 类：

- 主干网络：通常是一个用于抽取特征图的 FCN 网络，例如：ResNet, MobileNet。
- 模型颈部：通常是连接骨干网络和模型头部的组件，例如：ChannelMapper,FPN。
- 模型头部：用于特定任务的组件，例如：跟踪框预测。
- 损失函数：计算损失函数的部件，例如：FocalLoss, L1Loss。

13.1 增加一个新的主干网络

这里，我们以 MobileNet 为例来展示如何开发一个新组件。

13.1.1 1. 定义一个新主干网络（例如：MobileNet）

创建一个新文件 `mmtrack/models/backbones/mobilenet.py`

```
import torch.nn as nn
from mmcv.runner import BaseModule

from mmdet.models.builder import BACKBONES


```

(下页继续)

(续上页)

```
@BACKBONES.register_module()
class MobileNet(BaseModule):

    def __init__(self, arg1, arg2, *args, **kwargs):
        pass

    def forward(self, x):  # should return a tuple
        pass
```

13.1.2 2. 引进模块

你可以在 mmtrack/models/backbones/__init__.py 增加下面一行

```
from .mobilenet import MobileNet
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.models.backbones.mobilenet'],
    allow_failed_imports=False)
```

13.1.3 3. 更改原始 config 文件

```
model = dict(
    ...
    backbone=dict(
        type='MobileNet',
        arg1=xxx,
        arg2=xxx),
    ...
```

13.2 增加一个新的模型瓶颈

13.2.1 1. 定义一个模型瓶颈（例如：MyFPN）

创建一个新文件 mmtrack/models/necks/my_fpn.py

```
from mmcv.runner import BaseModule

```

(下页继续)

(续上页)

```
from mmdet.models.builder import NECKS

@NECKS.register_module()
class MyFPN(BaseModule):

    def __init__(self, arg1, arg2, *args, **kwargs):
        pass

    def forward(self, inputs):
        # implementation is ignored
        pass
```

13.2.2 2. 引进模块

你可以在 mmtrack/models/necks/__init__.py 增加下面一行

```
from .my_fpn import MyFPN
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.models.necks.my_fpn.py'],
    allow_failed_imports=False)
```

13.2.3 3. 更改原始 config 文件

```
neck=dict(
    type='MyFPN',
    arg1=xxx,
    arg2=xxx),
```

13.3 增加一个新的模型头部

13.3.1 1. 定义一个模型头部（例如：MyHead）

创建一个新文件 mmtrack/models/track_heads/my_head.py

```
from mmcv.runner import BaseModule

```

(下页继续)

(续上页)

```
from mmdet.models import HEADS

@HEADS.register_module()
class MyHead(BaseModule):

    def __init__(self, arg1, arg2, *args, **kwargs):
        pass

    def forward(self, inputs):
        # implementation is ignored
        pass
```

13.3.2 2. 引进模块

你可以在 mmtrack/models/track_heads/__init__.py 增加下面一行

```
from .my_head import MyHead
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.models.track_heads.my_head.py'],
    allow_failed_imports=False)
```

13.3.3 3. 更改原始 config 文件

```
track_head=dict(
    type='MyHead',
    arg1=xxx,
    arg2=xxx)
```

13.4 增加一个新的损失函数

详细请参考 增加新损失函数

CHAPTER 14

自定义训练配置

14.1 自定义优化设置

14.1.1 自定义 Pytorch 中的优化器

我们已经支持使用 Pytorch 所有的优化器，并且仅在 config 文件中更改设置即可使用。例如，如果你使用 ADAM，更改如下：

```
optimizer = dict(type='Adam', lr=0.0003, weight_decay=0.0001)
```

为了更改模型的学习率，使用者只需要更改 config 文件中的优化器的 lr 参数。使用者可以直接按照 Pytorch 的 API doc设置参数。

14.1.2 自定义自己实现的优化器

14.1.3 1. 定义一个新的优化器

一个自定义的优化器如下：

假定你增加的优化器为 MyOptimizer，它有参数 a，b，c。你需要建立一个新的文件 mmtrack/core/optimizer/my_optimizer.py。

```
from torch.optim import Optimizer
from mmcv.runner.optimizer import OPTIMIZERS
```

(下页继续)

(续上页)

```
@OPTIMIZERS.register_module()
class MyOptimizer(Optimizer):

    def __init__(self, a, b, c)
```

14.1.4 2. 将优化器加入注册器中

为了能够找到上述定义的模块，它应首先被引入到主命名空间中。我们提供两种方式来实现：

- 在文件 mmtrack/core/optimizer/__init__.py 中引入

新定义的 module 应被引入到 mmtrack/core/optimizer/__init__.py，以便注册器能够发现该新模块并添加它。

```
from .my_optimizer import MyOptimizer
```

- 在 config 文件中使用 custom_imports 来手动的引用它

```
custom_imports = dict(imports=['mmtrack.core.optimizer.my_optimizer.py'], allow_
↪failed_imports=False)
```

在项目开始阶段，模块 mmtrack.core.optimizer.my_optimizer.MyOptimizer 将会被引入，MyOptimizer 类会被自动注册。注意：我们引入的应该是只包含 MyOptimizer 的文件，而不是直接像这样 mmtrack.core.optimizer.my_optimizer.MyOptimizer 引入该类。

实际上使用者也可以将该模块定义在别的文件目录，只要该模块所在目录在 PYTHONPATH 里面能被找到。

14.1.5 3. 在 config 文件中指定优化器

你可以在 config 的 optimizer 区域使用 MyOptimizer。在 config 文件中，原始优化器定义如下：

```
optimizer = dict(type='SGD', lr=0.02, momentum=0.9, weight_decay=0.0001)
```

为了使用你自己的优化器，该区域可以更改如下：

```
optimizer = dict(type='MyOptimizer', a=a_value, b=b_value, c=c_value)
```

14.1.6 自定义优化器构建器

有的模型有一些用于模型优化的特定参数，例如：用于批归一化层的权值衰减。使用者可以通过自定义优化器构建器来调节这些参数。

```
from mmcv.utils import build_from_cfg

from mmcv.runner.optimizer import OPTIMIZER_BUILDERS, OPTIMIZERS
from mmtrack.utils import get_root_logger
from .my_optimizer import MyOptimizer


@OPTIMIZER_BUILDERS.register_module()
class MyOptimizerConstructor(object):

    def __init__(self, optimizer_cfg, paramwise_cfg=None):

    def __call__(self, model):

        return my_optimizer
```

默认的优化器构建器在此, 它可作为新优化器构建器的模板。

14.1.7 额外的设置

没有被优化器实现的技巧应该被优化器配置器 (例如：参数特定学习率) 或者钩子实现。我们列举了一些常用的可以稳定训练或者加速训练的设置。大家可以自由提出 PR、issue 来得到更多的设置。

- **使用梯度裁剪来稳定训练**：一些模型需要梯度裁剪来稳定训练过程。如下所示：

```
optimizer_config = dict(
    _delete_=True, grad_clip=dict(max_norm=35, norm_type=2))
```

如果你的 config 继承于基础 config, 并且已经设置了优化器基础 config, 你需要设置 `_delete=True` 来重写不必要的设置。详情请见 config 文档

- **使用动量调度器来加快模型收敛**。我们支持动量调度器来根据学习率改变模型的动量，使模型以更快的方式收敛。动量调度器通常和学习率调度器一起使用，例如：下面在 3D 检测中使用的 config 来加速模型收敛。更多细节请参考 CyclicLrUpdater and CyclicMomentumUpdater。

```
lr_config = dict(
    policy='cyclic',
    target_ratio=(10, 1e-4),
    cyclic_times=1,
    step_ratio_up=0.4,
```

(下页继续)

(续上页)

```
)
momentum_config = dict(
    policy='cyclic',
    target_ratio=(0.85 / 0.95, 1),
    cyclic_times=1,
    step_ratio_up=0.4,
)
```

14.2 自定义训练调度器

我们支持多种学习率调整策略，例如 CosineAnnealing 和 Poly。如下所示：

- Poly 调度器

```
lr_config = dict(policy='poly', power=0.9, min_lr=1e-4, by_epoch=False)
```

- CosineAnnealing 调度器

```
lr_config = dict(
    policy='CosineAnnealing',
    warmup='linear',
    warmup_iters=1000,
    warmup_ratio=1.0 / 10,
    min_lr_ratio=1e-5)
```

14.3 自定义工作流

工作流是一个包含（阶段名，迭代轮数）的列表，用来指定运行顺序和迭代轮数。默认设置如下：

```
workflow = [('train', 1)]
```

上述代码表示执行一轮训练阶段。有时，使用者可能想要检查模型在验证集上的一些度量指标 (例如：损失函数和准确度)。这种情况下，我们设置如下：

```
[('train', 1), ('val', 1)]
```

这样，经过一轮训练阶段后，将执行一次验证阶段。

注意

1. 模型的参数将不会在验证阶段更新。
2. config 中的关键字 total_epoch 是用来控制训练阶段轮数，不影响验证阶段。

3. 工作流 `[('train',1),('val',1)]` 和 `[('train',1)]` 将不会改变 `EvalHook`，因为 `EvalHook` 是被 `after_train_epoch` 调用，验证工作流仅影响在 `after_val_epoch` 中调用的钩子。因此，`[('train',1),('val',1)]` 和 `[('train',1)]` 唯一的不同就是 runner 将在每个训练循环结束后计算验证集上的损失函数。

14.4 自定义钩子

14.4.1 自定义自己实现的钩子

14.4.2 1. 实现一个新的钩子

在很多情况下，使用者需要实现一个新的钩子。MMTracking 支持在训练阶段自定义钩子。因此，使用者可以在 mmtrack 或者基于 mmtrack 的代码库中，通过修改训练 config 定义一个钩子。这里我们给出一个在 mmtrack 中创建一个钩子，并在 training 中使用它的范例：

```
from mmcv.runner import HOOKS, Hook


@HOOKS.register_module()
class MyHook(Hook):

    def __init__(self, a, b):
        pass

    def before_run(self, runner):
        pass

    def after_run(self, runner):
        pass

    def before_epoch(self, runner):
        pass

    def after_epoch(self, runner):
        pass

    def before_iter(self, runner):
        pass

    def after_iter(self, runner):
        pass
```

根据钩子的功能，用户需要指定在训练的每个阶段 `before_run`，`after_run`，`before_epoch`，

after_epoch, before_iter, and after_iter 钩子执行的事情。

14.4.3 2. 注册一个新的钩子

假定你需要注册的钩子为 MyHook，你可以在 mmtrack/core/utils/__init__.py 增加下面一行

```
from .my_hook import MyHook
```

或者，为了避免更改原始代码，你还可以在 config 文件中增加以下几行来实现：

```
custom_imports = dict(
    imports=['mmtrack.core.utils.my_hook'],
    allow_failed_imports=False)
```

14.4.4 3. 更改 config

```
custom_hooks = [
    dict(type='MyHook', a=a_value, b=b_value)
]
```

你也可以通过设置关键词 priority 为 NORMAL 或者 HIGHEST 来设置钩子的优先级:

```
custom_hooks = [
    dict(type='MyHook', a=a_value, b=b_value, priority='NORMAL')
]
```

在注册器中钩子的默认优先级为 NORMAL。

14.4.5 使用 MMCV 中已定义的钩子

如果钩子已经在 MMCV 中定义实现过，你可以直接在 config 文件中添加：

```
custom_hooks = [
    dict(type='MyHook', a=a_value, b=b_value, priority='NORMAL')
]
```

14.4.6 修改默认的钩子

有一些常用的钩子不是通过 custom_hooks 来注册，他们包括：

- log_config
- checkpoint_config
- evaluation
- lr_config
- optimizer_config
- momentum_config

在这些钩子中，只有日志钩子是 VERY_LOW 优先级，其余的优先级为 NORMAL。上述的教程已经包括如何修改 optimizer_config, momentum_config 和 lr_config。这里我们展示可以用 log_config、checkpoint_config 和 evaluation 做的事情。

14.4.7 模型保存钩子

MMCV 中的 runner 使用 checkpoint_config 来初始化模型保存钩子

```
checkpoint_config = dict(interval=1)
```

使用者可以设置 max_keep_ckpts 来仅保存一部分历史模型，可以设置 save_optimizer 来决定是否保存优化器参数。更多细节请参考模型保存钩子

14.4.8 日志钩子

log_config 包括多个日志钩子，并可以设置间隔时间。目前 MMCV 支持 WandbLoggerHook、MlflowLoggerHook、TensorboardLoggerHook。具体使用可以参考日志钩子

```
log_config = dict(
    interval=50,
    hooks=[
        dict(type='TextLoggerHook'),
        dict(type='TensorboardLoggerHook')
    ])
```

14.4.9 评估钩子

config 中的 evaluation 将被用来初始化评估钩子。除了 intertal、start 等关键词外，其他参数例如 metric 将被传入 dataset.evaluate()

```
evaluation = dict(interval=1, metric='bbox')
```

在 tools/目录下我们提供了许多有用的工具。

CHAPTER 15

MOT 测试时参数搜索

tools/analysis/mot/mot_param_search.py 脚本可以搜索 MOT 模型中跟踪器的参数。除了在配置文件上有所差别，该脚本的使用方法与 tools/test.py 脚本类似。

这里有一个示例来展示如何修改配置文件：

1. 定义所需的评测指标。

   比如你可以定义如下评测指标

```
search_metrics = ['MOTA', 'IDF1', 'FN', 'FP', 'IDs', 'MT', 'ML']
```

2. 定义需要被搜索的参数和数值

   假设你的跟踪器如下所示

```
model = dict(
    tracker=dict(
        type='BaseTracker',
        obj_score_thr=0.5,
        match_iou_thr=0.5
    )
)
```

   如果你想搜索这个跟踪器的参数，只需要将对应数值改成列表即可

```
model = dict(
    tracker=dict(
```

(下页继续)

(续上页)

```
        type='BaseTracker',
        obj_score_thr=[0.4, 0.5, 0.6],
        match_iou_thr=[0.4, 0.5, 0.6, 0.7]
    )
)
```

脚本将测试共 12 个案例，并记录相应测试结果。

CHAPTER 16

SiameseRPN++ 测试时参数搜索

`tools/analysis/sot/sot_siamrpn_param_search.py` 用来搜索 SiameseRPN++ 测试时的跟踪相关参数：`penalty_k`, `lr` 和 `window_influence`。你需要在参数解析器中传入前面每个参数的搜索范围。

在 UAV123 上的超参搜索范例：

```
./tools/analysis/sot/dist_sot_siamrpn_param_search.sh [${CONFIG_FILE}] [$GPUS] \
[--checkpoint ${CHECKPOINT}] [--log ${LOG_FILENAME}] [--eval ${EVAL}] \
[--penalty-k-range 0.01,0.22,0.05] [--lr-range 0.4,0.61,0.05] [--win-infu-range 0.01,
↪0.22,0.05]
```

在 OTB100 上的超参搜索范例：

```
./tools/analysis/sot/dist_sot_siamrpn_param_search.sh [${CONFIG_FILE}] [$GPUS] \
[--checkpoint ${CHECKPOINT}] [--log ${LOG_FILENAME}] [--eval ${EVAL}] \
[--penalty-k-range 0.3,0.45,0.02] [--lr-range 0.35,0.5,0.02] [--win-infu-range 0.46,0.
↪55,0.02]
```

在 VOT2018 上的超参搜索范例：

```
./tools/analysis/sot/dist_sot_siamrpn_param_search.sh [${CONFIG_FILE}] [$GPUS] \
[--checkpoint ${CHECKPOINT}] [--log ${LOG_FILENAME}] [--eval ${EVAL}] \
[--penalty-k-range 0.01,0.31,0.05] [--lr-range 0.2,0.51,0.05] [--win-infu-range 0.3,0.
↪56,0.05]
```

CHAPTER 17

日志分析

tools/analysis/analyze_logs.py 脚本可以根据训练日志文件绘制损失函数以及 mAP 曲线。

```
python tools/analysis/analyze_logs.py plot_curve [--keys ${KEYS}] [--title ${TITLE}]␣
↪[--legend ${LEGEND}] [--backend ${BACKEND}] [--style ${STYLE}] [--out ${OUT_FILE}]
```

几个例子:

- 绘制某次运行时的分类损失函数。

```
python tools/analysis/analyze_logs.py plot_curve log.json --keys loss_cls --
↪legend loss_cls
```

- 绘制某次运行时的分类以及回归损失函数，并且保存成 pdf 文件。

```
python tools/analysis/analyze_logs.py plot_curve log.json --keys loss_cls loss_
↪bbox --out losses.pdf
```

- 在同一张图中比较两次运行的 bbox mAP。

```
python tools/analysis/analyze_logs.py plot_curve log1.json log2.json --keys bbox_
↪mAP --legend run1 run2
```

- 计算平均运行速度

```
python tools/analysis/analyze_logs.py cal_train_time log.json [--include-outliers]
```

输出如下所示：

```
-----Analyze train time of work_dirs/some_exp/20190611_192040.log.json-----
slowest epoch 11, average time is 1.2024
fastest epoch 1, average time is 1.1909
time std over epochs is 0.0028
average iter time: 1.1959 s/iter
```

CHAPTER 18

模型转换

18.1 发布模型

tools/analysis/publish_model.py 脚本可以帮助用户发布模型。

在将模型上传到 AWS 之前，你可能想要做以下事情：

1. 将模型参数转化成 CPU 张量
2. 删除优化器状态参数
3. 计算模型权重文件的哈希值并将其添加进文件名。

```
python tools/analysis/publish_model.py ${INPUT_FILENAME} ${OUTPUT_FILENAME}
```

比如：

```
python tools/analysis/publish_model.py work_dirs/dff_faster_rcnn_r101_dc5_1x_
↪imagenetvid/latest.pth dff_faster_rcnn_r101_dc5_1x_imagenetvid.pth
```

最后输出的文件名为 dff_faster_rcnn_r101_dc5_1x_imagenetvid_20201230-{hash id}.pth。

CHAPTER 19

其它有用的工具脚本

19.1 输出完整的配置

tools/analysis/print_config.py 脚本可以输出完整的配置，包括文件中导入的配置。

```
python tools/analysis/print_config.py ${CONFIG} [-h] [--options ${OPTIONS [OPTIONS...
↪]}]
```

CHAPTER 20

更新日志

20.1 v0.6.0 (30/07/2021)

20.1.1 亮点

- 修复三个任务的训练问题 (#219), (#221)

20.1.2 新特性

- 支持多目标跟踪任务的错误分析可视化 (#212)

20.1.3 问题修复

- 修复单目标跟踪演示中的一个问题 (#213)

20.1.4 提升

- 使用 MMCV 中的注册器 (#220)
- 增加重识别任务训练教程 (#210)
- 修改单目标跟踪输出字典的键值 (#223)
- 增加四篇中文文档 install.md, quick_run.md, model_zoo.md, dataset.md (#205), (#214)

20.2 v0.5.3 (01/07/2021)

20.2.1 新特性

- 支持重识别任务的训练 (#177, #179, #180, #181)
- 支持 MIM (#158)

20.2.2 问题修复

- 修复评测钩子 (#176)
- 修复视频目标检测配置中的错字 (#171)

20.2.3 提升

- 重构 nms 配置 (#167)

20.3 v0.5.2 (03/06/2021)

20.3.1 提升

- 修复错字 (#104, #121, #145)
- 增加会议引用 (#111)
- 更新 CONTRIBUTING 链接到 mmcv (#112)
- 调整 mmcv 中的更新 (FP16Hook) (#114, #119)
- 添加了指向其他代码库的 bibtex 和链接 (#122)
- 添加 docker 文件 (#124)
- 使用 mmcv 中的 `collect_env` (#129)

- 增加和更新中文教程 (#135, #147, #148)

20.4 v0.5.1 (01/02/2021)

20.4.1 问题修复

- 修复重识别模型权重文件的导入 (#80)
- 修复 `track_result` 中的空张量 (#86)
- 修复多目标跟踪演示脚本中的 `wait_time` (#92)

20.4.2 提升

- 支持 DeepSORT 使用单阶段检测器 (#100)

20.5 v0.5.0 (04/01/2021)

20.5.1 亮点

- MMTracking 已经发布!

20.5.2 新特性

- 支持的视频目标检测方法: DFF, FGFA, SELSA
- 支持的多目标跟踪方法: SORT/DeepSORT, Tracktor
- 支持的单目标跟踪方法: SiameseRPN++

CHAPTER 21

English

CHAPTER 22

简体中文

CHAPTER 23

mmtrack.apis

mmtrack.apis.**inference_mot**(*model*, *img*, *frame_id*)

Inference image(s) with the mot model.

参数

- **model** (*nn.Module*) –The loaded mot model.
- **img** (*str | ndarray*) –Either image name or loaded image.
- **frame_id** (*int*) –frame id.

返回 ndarray]: The tracking results.

返回类型 dict[str

mmtrack.apis.**inference_sot**(*model*, *image*, *init_bbox*, *frame_id*)

Inference image with the single object tracker.

参数

- **model** (*nn.Module*) –The loaded tracker.
- **image** (*ndarray*) –Loaded images.
- **init_bbox** (*ndarray*) –The target needs to be tracked.
- **frame_id** (*int*) –frame id.

返回 ndarray]: The tracking results.

返回类型 dict[str

mmtrack.apis.**inference_vid**(*model*, *image*, *frame_id*, *ref_img_sampler={'frame_stride': 10, 'num_left_ref_imgs': 10}*)

Inference image with the video object detector.

参数

- **model** (*nn.Module*) –The loaded detector.
- **image** (*ndarray*) –Loaded images.
- **frame_id** (*int*) –Frame id.
- **ref_img_sampler** (*dict*) –The configuration for sampling reference images. Only used under video detector of fgfa style. Defaults to dict(frame_stride=2, num_left_ref_imgs=10).

返回 ndarray]: The detection results.

返回类型 dict[str

mmtrack.apis.**init_model**(*config*, *checkpoint=None*, *device='cuda:0'*, *cfg_options=None*, *verbose_init_params=False*)

Initialize a model from config file.

参数

- **config** (str or mmcv.Config) –Config file path or the config object.
- **checkpoint** (*str, optional*) –Checkpoint path. Default as None.
- **cfg_options** (*dict, optional*) –Options to override some settings in the used config. Default to None.
- **verbose_init_params** (*bool, optional*) –Whether to print the information of initialized parameters to the console. Default to False.

返回 The constructed detector.

返回类型 nn.Module

mmtrack.apis.**init_random_seed**(*seed=None*, *device='cuda'*)

Initialize random seed.

If the seed is not set, the seed will be automatically randomized, and then broadcast to all processes to prevent some potential bugs. :param seed: The seed. Default to None. :type seed: int, Optional :param device: The device where the seed will be put on.

Default to ‘cuda’.

返回 Seed to be used.

返回类型 int

mmtrack.apis.**multi_gpu_test**(*model*, *data_loader*, *tmpdir=None*, *gpu_collect=False*)

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. ‘gpu_collect=True’ is not supported for now.

参数

- **model** (*nn.Module*) –Model to be tested.
- **data_loader** (*nn.Dataloader*) –Pytorch data loader.
- **tmpdir** (*str*) –Path of directory to save the temporary results from different gpus under cpu mode. Defaults to None.
- **gpu_collect** (*bool*) –Option to use either gpu or cpu to collect results. Defaults to False.

返回 The prediction results.

返回类型 dict[str, list]

mmtrack.apis.**single_gpu_test**(*model*, *data_loader*, *show=False*, *out_dir=None*, *fps=3*, *show_score_thr=0.3*)

Test model with single gpu.

参数

- **model** (*nn.Module*) –Model to be tested.
- **data_loader** (*nn.Dataloader*) –Pytorch data loader.
- **show** (*bool, optional*) –If True, visualize the prediction results. Defaults to False.
- **out_dir** (*str, optional*) –Path of directory to save the visualization results. Defaults to None.
- **fps** (*int, optional*) –FPS of the output video. Defaults to 3.
- **show_score_thr** (*float, optional*) –The score threshold of visualization (Only used in VID for now). Defaults to 0.3.

返回 The prediction results.

返回类型 dict[str, list]

mmtrack.apis.**train_model**(*model*, *dataset*, *cfg*, *distributed=False*, *validate=False*, *timestamp=None*, *meta=None*)

Train model entry function.

参数

- **model** (*nn.Module*) –The model to be trained.
- **dataset** (Dataset) –Train dataset.
- **cfg** (*dict*) –The config dict for training.

- **distributed** (*bool*) –Whether to use distributed training. Default: False.
- **validate** (*bool*) –Whether to do evaluation. Default: False.
- **timestamp** (*str | None*) –Local time for runner. Default: None.
- **meta** (*dict | None*) –Meta dict to record some important information. Default: None

CHAPTER 24

mmtrack.core

24.1 anchor

class mmtrack.core.anchor.**SiameseRPNAnchorGenerator**(*strides*, **args*, ***kwargs*)

Anchor generator for siamese rpn.

Please refer to *mmdet/core/anchor/anchor_generator.py:AnchorGenerator* for detailed docstring.

gen_2d_hanning_windows(*featmap_sizes*, *device='cuda'*)

Generate 2D hanning window.

参数

- **featmap_sizes** (*list[torch.size]*) –List of torch.size recording the resolution (height, width) of the multi-level feature maps.
- **device** (*str*) –Device the tensor will be put on. Defaults to 'cuda'.

返回 List of 2D hanning window with shape (num_base_anchors[i] * featmap_sizes[i][0] * featmap_sizes[i][1]).

返回类型 list[Tensor]

gen_single_level_base_anchors(*base_size*, *scales*, *ratios*, *center=None*)

Generate base anchors of a single level feature map.

参数

- **base_size** (*int | float*) –Basic size of an anchor.

- **scales** (*torch.Tensor*) –Scales of the anchor.
- **ratios** (*torch.Tensor*) –The ratio between between the height and width of anchors in a single level.
- **center** (*tuple[float], optional*) –The center of the base anchor related to a single feature grid. Defaults to None.

返回 Anchors of one spatial location in a single level feature map in [tl_x, tl_y, br_x, br_y] format.

返回类型 torch.Tensor

24.2 evaluation

class mmtrack.core.evaluation.**DistEvalHook**(*dataloader: torch.utils.data.dataloader.DataLoader*, *start: Optional[int] = None*, *interval: int = 1*, *by_epoch: bool = True*, *save_best: Optional[str] = None*, *rule: Optional[str] = None*, *test_fn: Optional[Callable] = None*, *greater_keys: Optional[List[str]] = None*, *less_keys: Optional[List[str]] = None*, *broadcast_bn_buffer: bool = True*, *tmpdir: Optional[str] = None*, *gpu_collect: bool = False*, *out_dir: Optional[str] = None*, *file_client_args: Optional[dict] = None*, ***eval_kwargs*)

Please refer to *mmcv.runner.hooks.evaluation.py:DistEvalHook* for detailed docstring.

class mmtrack.core.evaluation.**EvalHook**(*dataloader: torch.utils.data.dataloader.DataLoader*, *start: Optional[int] = None*, *interval: int = 1*, *by_epoch: bool = True*, *save_best: Optional[str] = None*, *rule: Optional[str] = None*, *test_fn: Optional[Callable] = None*, *greater_keys: Optional[List[str]] = None*, *less_keys: Optional[List[str]] = None*, *out_dir: Optional[str] = None*, *file_client_args: Optional[dict] = None*, ***eval_kwargs*)

Please refer to *mmcv.runner.hooks.evaluation.py:EvalHook* for detailed docstring.

mmtrack.core.evaluation.**bbox2region**(*bbox*)

Convert bbox to Rectangle or Polygon Class object.

参数 **bbox** (*ndarray*) –the format of rectangle bbox is (x1, y1, w, h); the format of polygon is (x1, y1, x2, y2, ···).

返回 Rectangle or Polygon Class object.

mmtrack.core.evaluation.**eval_mot**(*results*, *annotations*, *logger=None*, *classes=None*, *iou_thr=0.5*, *ignore_iof_thr=0.5*, *ignore_by_classes=False*, *nproc=4*)

Evaluation CLEAR MOT metrics.

参数

- **results** (*list[list[list[ndarray]]]*) –The first list indicates videos, The second list indicates images. The third list indicates categories. The ndarray indicates the tracking results.
- **annotations** (*list[list[dict]]*) –The first list indicates videos, The second list indicates images. The third list indicates the annotations of each video. Keys of annotations are
  - *bboxes*: numpy array of shape (n, 4)
  - *labels*: numpy array of shape (n,)
  - *instance_ids*: numpy array of shape (n,)
  - *bboxes_ignore* (optional): numpy array of shape (k, 4)
  - *labels_ignore* (optional): numpy array of shape (k,)
- **logger** (*logging.Logger | str | None, optional*) –The way to print the evaluation results. Defaults to None.
- **classes** (*list, optional*) –Classes in the dataset. Defaults to None.
- **iou_thr** (*float, optional*) –IoU threshold for evaluation. Defaults to 0.5.
- **ignore_iof_thr** (*float, optional*) –Iof threshold to ignore results. Defaults to 0.5.
- **ignore_by_classes** (*bool, optional*) –Whether ignore the results by classes or not. Defaults to False.
- **nproc** (*int, optional*) –Number of the processes. Defaults to 4.

返回 Evaluation results.

返回类型 dict[str, float]

mmtrack.core.evaluation.**eval_sot_accuracy_robustness**(*results*, *annotations*, *burnin=10*, *ignore_unknown=True*, *videos_wh=None*)

Calculate accuracy and robustness over all tracking sequences.

参数

- **results** (*list[list[ndarray]]*) –The first list contains the tracking results of each video. The second list contains the tracking results of each frame in one video. The ndarray have two cases:

  - **bbox: denotes the normal tracking box in [x1, y1, w, h]** format.
  - **special tracking state: [0] denotes the unknown state,** namely the skipping frame after failure, [1] denotes the initialized state, and [2] denotes the failed state.

- **annotations** (*list[ndarray]*) –The list contains the gt_bboxes of each video. The ndarray is gt_bboxes of one video. It' s in (N, 4) shape. Each bbox is in (x1, y1, w, h) format.
- **burnin** –number of frames that have to be ignored after the re-initialization when calculating accuracy. Default is 10.
- **ignore_unknown** (*bool*) –whether ignore the skipping frames after failures when calculating accuracy. Default is True.
- **videos_wh** (*list[tuple(width, height), ...]*) –The list contains the width and height of each video. Default is None.

返回 float}: accuracy and robustness in EAO evaluation metric.

返回类型 dict{str

mmtrack.core.evaluation.**eval_sot_eao**(*results*, *annotations*, *interval=[100, 356]*, *videos_wh=None*)

Calculate EAO socre over all tracking sequences.

参数

- **results** (*list[list[ndarray]]*) –The first list contains the tracking results of each video. The second list contains the tracking results of each frame in one video. The ndarray have two cases:
  - **bbox: denotes the normal tracking box in [x1, y1, w, h]** format.
  - **special tracking state: [0] denotes the unknown state,** namely the skipping frame after failure, [1] denotes the initialized state, and [2] denotes the failed state.
- **annotations** (*list[ndarray]*) –The list contains the gt_bboxes of each video. The ndarray is gt_bboxes of one video. It' s in (N, 4) shape. Each bbox is in (x1, y1, w, h) format.
- **interval** –an specified interval in EAO curve used to calculate the EAO score. There are different settings in different VOT challenge. Default is VOT2018 setting: [100, 356].
- **videos_wh** (*list[tuple(width, height), ...]*) –The list contains the width and height of each video. Default is None.

返回 EAO score in EAO evaluation metric.

返回类型 dict[str, float]

mmtrack.core.evaluation.**eval_sot_ope**(*results*, *annotations*, *visible_infos=None*)

Evaluation in OPE protocol.

参数

- **results** (*list[list[ndarray]]*) –The first list contains the tracking results of each video. The second list contains the tracking results of each frame in one video. The ndarray denotes the tracking box in [tl_x, tl_y, br_x, br_y] format.
- **annotations** (*list[ndarray]*) –The list contains the bbox annotations of each video. The ndarray is gt_bboxes of one video. It' s in (N, 4) shape. Each bbox is in (x1, y1, x2, y2) format.
- **visible_infos** (*list[ndarray] | None*) –If not None, the list contains the visible information of each video. The ndarray is visibility (with bool type) of object in one video. It' s in (N,) shape. Default to None.

返回 OPE style evaluation metric (i.e. success, norm precision and precision).

返回类型 dict[str, float]

mmtrack.core.evaluation.**eval_vis**(*test_results*, *vis_anns*, *logger=None*)

Evaluation on VIS metrics.

参数

- **test_results** (*dict(list[dict])*) –Testing results of the VIS dataset.
- **vis_anns** (*dict(list[dict])*) –The annotation in the format of YouTube-VIS.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.

返回 Evaluation results.

返回类型 dict[str, float]

24.3 motion

mmtrack.core.motion.**flow_warp_feats**(*x*, *flow*)

Use flow to warp feature map.

参数

- **x** (*Tensor*) –of shape (N, C, H_x, W_x).
- **flow** (*Tensor*) –of shape (N, C, H_f, W_f).

返回 The warpped feature map with shape (N, C, H_x, W_x).

返回类型 Tensor

24.4 optimizer

class mmtrack.core.optimizer.**SiameseRPNFp16OptimizerHook**(*backbone_start_train_epoch*, *backbone_train_layers*, ***kwargs*)

FP16Optimizer hook for siamese rpn.

参数

- **backbone_start_train_epoch** (*int*) –Start to train the backbone at *backbone_start_train_epoch*-th epoch. Note the epoch in this class counts from 0, while the epoch in the log file counts from 1.
- **backbone_train_layers** (*list(str)*) –List of str denoting the stages needed be trained in backbone.

before_train_epoch(*runner*)

If *runner.epoch* >= *self.backbone_start_train_epoch*, start to train the backbone.

class mmtrack.core.optimizer.**SiameseRPNLrUpdaterHook**(*lr_configs=[{'type': 'step', 'start_lr_factor': 0.2, 'end_lr_factor': 1.0, 'end_epoch': 5}, {'type': 'log', 'start_lr_factor': 1.0, 'end_lr_factor': 0.1, 'end_epoch': 20}]*, ***kwargs*)

Learning rate updater for siamese rpn.

参数 **lr_configs** (*list[dict]*) –List of dict where each dict denotes the configuration of specifical learning rate updater and must have ‘type’.

get_lr(*runner*, *base_lr*)

Get a specifical learning rate for each epoch.

class mmtrack.core.optimizer.**SiameseRPNOptimizerHook**(*backbone_start_train_epoch*, *backbone_train_layers*, ***kwargs*)

Optimizer hook for siamese rpn.

参数

- **backbone_start_train_epoch** (*int*) –Start to train the backbone at *backbone_start_train_epoch*-th epoch. Note the epoch in this class counts from 0, while the epoch in the log file counts from 1.
- **backbone_train_layers** (*list(str)*) –List of str denoting the stages needed be trained in backbone.

before_train_epoch(*runner*)

If *runner.epoch* >= *self.backbone_start_train_epoch*, start to train the backbone.

24.5 track

mmtrack.core.track.**depthwise_correlation**(*x*, *kernel*)

Depthwise cross correlation.

This function is proposed in SiamRPN++.

参数

- **x** (*Tensor*) –of shape (N, C, H_x, W_x).
- **kernel** (*Tensor*) –of shape (N, C, H_k, W_k).

返回 of shape (N, C, H_o, W_o). H_o = H_x - H_k + 1. So does W_o.

返回类型 Tensor

mmtrack.core.track.**embed_similarity**(*key_embeds*, *ref_embeds*, *method='dot_product'*, *temperature=-1*)

Calculate feature similarity from embeddings.

参数

- **key_embeds** (*Tensor*) –Shape (N1, C).
- **ref_embeds** (*Tensor*) –Shape (N2, C).
- **method** (*str, optional*) –Method to calculate the similarity, options are ‘dot_product’ and ‘cosine’. Defaults to ‘dot_product’.
- **temperature** (*int, optional*) –Softmax temperature. Defaults to -1.

返回 Similarity matrix of shape (N1, N2).

返回类型 Tensor

mmtrack.core.track.**imrenormalize**(*img*, *img_norm_cfg*, *new_img_norm_cfg*)

Re-normalize the image.

参数

- **img** (*Tensor | ndarray*) –Input image. If the input is a Tensor, the shape is (1, C, H, W). If the input is a ndarray, the shape is (H, W, C).
- **img_norm_cfg** (*dict*) –Original configuration for the normalization.
- **new_img_norm_cfg** (*dict*) –New configuration for the normalization.

返回 Output image with the same type and shape of the input.

返回类型 Tensor | ndarray

mmtrack.core.track.**interpolate_tracks**(*tracks*, *min_num_frames=5*, *max_num_frames=20*)

Interpolate tracks linearly to make tracks more complete.

This function is proposed in "ByteTrack: Multi-Object Tracking by Associating Every Detection Box." **'ByteTrack<https://arxiv.org/abs/2110.06864>'_**.

参数

- **tracks** (*ndarray*) –With shape (N, 7). Each row denotes (frame_id, track_id, x1, y1, x2, y2, score).
- **min_num_frames** (*int, optional*) –The minimum length of a track that will be interpolated. Defaults to 5.
- **max_num_frames** (*int, optional*) –The maximum disconnected length in a track. Defaults to 20.

返回

The interpolated tracks with shape (N, 7). Each row denotes (frame_id, track_id, x1, y1, x2, y2, score)

返回类型 ndarray

mmtrack.core.track.**outs2results**(*bboxes=None*, *labels=None*, *masks=None*, *ids=None*, *num_classes=None*, ***kwargs*)

Convert tracking/detection results to a list of numpy arrays.

参数

- **bboxes** (*torch.Tensor | np.ndarray*) –shape (n, 5)
- **labels** (*torch.Tensor | np.ndarray*) –shape (n,)
- **masks** (*torch.Tensor | np.ndarray*) –shape (n, h, w)
- **ids** (*torch.Tensor | np.ndarray*) –shape (n,)
- **num_classes** (*int*) –class number, not including background class

返回

list(ndarray) | list[list[np.ndarray]]]: tracking/detection results of each class. It may contain keys as belows:

- **bbox_results (list[np.ndarray]): Each list denotes bboxes of one** category.
- **mask_results (list[list[np.ndarray]]): Each outer list denotes masks** of one category. Each inner list denotes one mask belonging to the category. Each mask has shape (h, w).

返回类型 dict[str

mmtrack.core.track.**results2outs**(*bbox_results=None*, *mask_results=None*, *mask_shape=None*, ***kwargs*)

Restore the results (list of results of each category) into the results of the model forward.

参数

- **bbox_results** (*list[np.ndarray]*) –Each list denotes bboxes of one category.
- **mask_results** (*list[list[np.ndarray]]*) –Each outer list denotes masks of one category. Each inner list denotes one mask belonging to the category. Each mask has shape (h, w).
- **mask_shape** (*tuple[int]*) –The shape (h, w) of mask.

返回

tracking results of each class. It may contain keys as belows:

- bboxes (np.ndarray): shape (n, 5)
- labels (np.ndarray): shape (n,)
- masks (np.ndarray): shape (n, h, w)
- ids (np.ndarray): shape (n,)

返回类型 tuple

24.6 utils

mmtrack.core.utils.**crop_image**(*image*, *crop_region*, *crop_size*, *padding=(0, 0, 0)*)

Crop image based on *crop_region* and *crop_size*.

参数

- **image** (*ndarray*) –of shape (H, W, 3).
- **crop_region** (*ndarray*) –of shape (4,) in [x1, y1, x2, y2] format.
- **crop_size** (*int*) –Crop size.
- **padding** (*tuple | ndarray*) –of shape (3,) denoting the padding values.

返回 Cropped image of shape (crop_size, crop_size, 3).

返回类型 ndarray

mmtrack.core.utils.**imshow_mot_errors**(**args*, *backend='cv2'*, ***kwargs*)

Show the wrong tracks on the input image.

参数 **backend** (*str, optional*) –Backend of visualization. Defaults to ‘cv2’.

mmtrack.core.utils.**imshow_tracks**(**args*, *backend='cv2'*, ***kwargs*)

Show the tracks on the input image.

CHAPTER 25

mmtrack.datasets

25.1 datasets

class mmtrack.datasets.**BaseSOTDataset**(*img_prefix*, *pipeline*, *split*, *ann_file=None*, *test_mode=False*, *bbox_min_size=0*, *only_eval_visible=False*, *file_client_args={'backend': 'disk'}*, ***kwargs*)

Dataset of single object tracking. The dataset can both support training and testing mode.

参数

- **img_prefix** (*str*) –Prefix in the paths of image files.
- **pipeline** (*list[dict]*) –Processing pipeline.
- **split** (*str*) –Dataset split.
- **ann_file** (*str, optional*) –The file contains data information. It will be loaded and parsed in the *self.load_data_infos* function.
- **test_mode** (*bool, optional*) –Default to False.
- **bbox_min_size** (*int, optional*) –Only bounding boxes whose sizes are larger than *bbox_min_size* can be regarded as valid. Default to 0.
- **only_eval_visible** (*bool, optional*) –Whether to only evaluate frames where object are visible. Default to False.
- **file_client_args** (*dict, optional*) –Arguments to instantiate a FileClient. Default: dict(backend='disk').

evaluate(*results*, *metric=['track']*, *logger=None*)

Default evaluation standard is OPE.

参数

- **results** (*dict(list[ndarray])*) –tracking results. The ndarray is in (x1, y1, x2, y2, score) format.
- **metric** (*list, optional*) –defaults to ['track'].
- **logger** (*logging.Logger | str | None, optional*) –defaults to None.

get_ann_infos_from_video(*video_ind*)

Get annotation information in a video.

参数 **video_ind** (*int*) –video index

返回

{ 'bboxes' : ndarray in (N, 4) shape, 'bboxes_isvalid' : ndarray, 'visible' :ndarray}. The annotation information in some datasets may contain 'visible_ratio' . The bbox is in (x1, y1, x2, y2) format.

返回类型 dict

get_bboxes_from_video(*video_ind*)

Get bboxes annotation about the instance in a video.

参数 **video_ind** (*int*) –video index

返回

in [N, 4] shape. The N is the number of bbox and the bbox is in (x, y, w, h) format.

返回类型 ndarray

get_img_infos_from_video(*video_ind*)

Get image information in a video.

参数 **video_ind** (*int*) –video index

返回 { 'filename' : list[str], 'frame_ids' :ndarray, 'video_id' :int}

返回类型 dict

get_len_per_video(*video_ind*)

Get the number of frames in a video.

get_visibility_from_video(*video_ind*)

Get the visible information of instance in a video.

load_as_video

The self.data_info is a list, which the length is the number of videos. The default content is in the following format: [

{ ‘video_path’: the video path ‘ann_path’: the annotation path ‘start_frame_id’: the starting frame ID number contained in

the image name

‘end_frame_id’: the ending frame ID number contained in the image name

‘framename_template’: the template of image name

]

pre_pipeline(*results*)

Prepare results dict for pipeline.

The following keys in dict will be called in the subsequent pipeline.

prepare_test_data(*video_ind*, *frame_ind*)

Get testing data of one frame. We parse one video, get one frame from it and pass the frame information to the pipeline.

参数

- **video_ind** (*int*) –video index
- **frame_ind** (*int*) –frame index

返回 testing data of one frame.

返回类型 dict

prepare_train_data(*video_ind*)

Get training data sampled from some videos. We firstly sample two videos from the dataset and then parse the data information. The first operation in the training pipeline is frames sampling.

参数 **video_ind** (*int*) –video index

返回 training data pairs, triplets or groups.

返回类型 dict

class mmtrack.datasets.**CocoVID**(**args: Any*, ***kwargs: Any*)

Inherit official COCO class in order to parse the annotations of bbox- related video tasks.

参数

- **annotation_file** (*str*) –location of annotation file. Defaults to None.
- **load_img_as_vid** (*bool*) –If True, convert image data to video data, which means each image is converted to a video. Defaults to False.

convert_img_to_vid(*dataset*)

Convert image data to video data.

createIndex()

Create index.

get_img_ids_from_ins_id(*insId*)

Get image ids from given instance id.

参数 **insId** (*int*) –The given instance id.

返回 Image ids of given instance id.

返回类型 list[int]

get_img_ids_from_vid(*vidId*)

Get image ids from given video id.

参数 **vidId** (*int*) –The given video id.

返回 Image ids of given video id.

返回类型 list[int]

get_ins_ids_from_vid(*vidId*)

Get instance ids from given video id.

参数 **vidId** (*int*) –The given video id.

返回 Instance ids of given video id.

返回类型 list[int]

get_vid_ids(*vidIds=[]*)

Get video ids that satisfy given filter conditions.

Default return all video ids.

参数 **vidIds** (*list[int]*) –The given video ids. Defaults to [].

返回 Video ids.

返回类型 list[int]

load_vids(*ids=[]*)

Get video information of given video ids.

Default return all videos information.

参数 **ids** (*list[int]*) –The given video ids. Defaults to [].

返回 List of video information.

返回类型 list[dict]

class mmtrack.datasets.**CocoVideoDataset**(*load_as_video=True*, *key_img_sampler={'interval': 1}*, *ref_img_sampler={'filter_key_img': True, 'frame_range': 10, 'method': 'uniform', 'num_ref_imgs': 1, 'return_key_img': True, 'stride': 1}*, *test_load_ann=False*, **args*, ***kwargs*)

Base coco video dataset for VID, MOT and SOT tasks.

参数

- **load_as_video** (*bool*) –If True, using COCOVID class to load dataset, otherwise, using COCO class. Default: True.
- **key_img_sampler** (*dict*) –Configuration of sampling key images.
- **ref_img_sampler** (*dict*) –Configuration of sampling ref images.
- **test_load_ann** (*bool*) –If True, loading annotations during testing, otherwise, not loading. Default: False.

evaluate(*results*, *metric=['bbox', 'track']*, *logger=None*, *bbox_kwargs={'classwise': False, 'iou_thrs': None, 'metric_items': None, 'proposal_nums': (100, 300, 1000)}*, *track_kwargs={'ignore_by_classes': False, 'ignore_iof_thr': 0.5, 'iou_thr': 0.5, 'nproc': 4}*)

Evaluation in COCO protocol and CLEAR MOT metric (e.g. MOTA, IDF1).

参数

- **results** (*dict*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated. Options are ‘bbox’, ‘segm’, ‘track’.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.
- **bbox_kwargs** (*dict*) –Configuration for COCO styple evaluation.
- **track_kwargs** (*dict*) –Configuration for CLEAR MOT evaluation.

返回 COCO style and CLEAR MOT evaluation metric.

返回类型 dict[str, float]

get_ann_info(*img_info*)

Get COCO annotations by the information of image.

参数 **img_info** (*int*) –Information of image.

返回 Annotation information of *img_info*.

返回类型 dict

key_img_sampling(*img_ids*, *interval=1*)

Sampling key images.

load_annotations(*ann_file*)

Load annotations from COCO/COCOVID style annotation file.

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

返回 Annotation information from COCO/COCOVID api.

返回类型 list[dict]

load_video_anns(*ann_file*)

Load annotations from COCOVID style annotation file.

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

返回 Annotation information from COCOVID api.

返回类型 list[dict]

prepare_data(*idx*)

Get data and annotations after pipeline.

参数 **idx** (*int*) –Index of data.

返回 Data and annotations after pipeline with new keys introduced by pipeline.

返回类型 dict

prepare_results(*img_info*)

Prepare results for image (e.g. the annotation information, ···).

prepare_test_img(*idx*)

Get testing data after pipeline.

参数 **idx** (*int*) –Index of data.

返回 Testing data after pipeline with new keys intorduced by pipeline.

返回类型 dict

prepare_train_img(*idx*)

Get training data and annotations after pipeline.

参数 **idx** (*int*) –Index of data.

返回 Training data and annotations after pipeline with new keys introduced by pipeline.

返回类型 dict

ref_img_sampling(*img_info*, *frame_range*, *stride=1*, *num_ref_imgs=1*, *filter_key_img=True*, *method='uniform'*, *return_key_img=True*)

Sampling reference frames in the same video for key frame.

参数

- **img_info** (*dict*) –The information of key frame.

- **frame_range** (*List(int) | int*) –The sampling range of reference frames in the same video for key frame.
- **stride** (*int*) –The sampling frame stride when sampling reference images. Default: 1.
- **num_ref_imgs** (*int*) –The number of sampled reference images. Default: 1.
- **filter_key_img** (*bool*) –If False, the key image will be in the sampling reference candidates, otherwise, it is exclude. Default: True.
- **method** (*str*) –The sampling method. Options are 'uniform', 'bilateral_uniform', 'test_with_adaptive_stride', 'test_with_fix_stride'. 'uniform' denotes reference images are randomly sampled from the nearby frames of key frame. 'bilateral_uniform' denotes reference images are randomly sampled from the two sides of the nearby frames of key frame. 'test_with_adaptive_stride' is only used in testing, and denotes the sampling frame stride is equal to (video length / the number of reference images). test_with_fix_stride is only used in testing with sampling frame stride equalling to *stride*. Default: 'uniform'.
- **return_key_img** (*bool*) –If True, the information of key frame is returned, otherwise, not returned. Default: True.

返回 *img_info* and the reference images information or only the reference images information.

返回类型 list(dict)

class mmtrack.datasets.**DanceTrackDataset**(*visibility_thr=-1*, *interpolate_tracks_cfg=None*, *detection_file=None*, **args*, ***kwargs*)

Dataset for DanceTrack: https://github.com/DanceTrack/DanceTrack.

Most content is inherited from MOTChallengeDataset.

get_benchmark_and_eval_split()

Get benchmark and dataset split to evaluate.

Get benchmark from upeper/lower-case image prefix and the dataset split to evaluate.

返回 The first string denotes the type of dataset. The second string denots the split of the dataset to eval.

返回类型 tuple(string)

class mmtrack.datasets.**GOT10kDataset**(**args*, ***kwargs*)

GOT10k Dataset of single object tracking.

The dataset can both support training and testing mode.

format_results(*results*, *resfile_path=None*, *logger=None*)

Format the results to txts (standard format for GOT10k Challenge).

参数

- **results** (*dict(list[ndarray])*) –Testing results of the dataset.

- **resfile_path** (*str*) –Path to save the formatted results. Defaults to None.
- **logger** (*logging.Logger | str | None, optional*) –defaults to None.

get_visibility_from_video(*video_ind*)

Get the visible information of instance in a video.

load_data_infos(*split='train'*)

Load dataset information.

参数 **split** (*str, optional*) –the split of dataset. Defaults to ‘train’.

返回

the length of the list is the number of videos. The

inner dict is in the following format:

{ ‘video_path’: the video path ‘ann_path’: the annotation path ‘start_frame_id’ : the starting frame number contained

in the image name

’end_frame_id’: the ending frame number contained in the image name

’framename_template’: the template of image name

}

返回类型 list[dict]

prepare_test_data(*video_ind*, *frame_ind*)

Get testing data of one frame. We parse one video, get one frame from it and pass the frame information to the pipeline.

参数

- **video_ind** (*int*) –video index
- **frame_ind** (*int*) –frame index

返回 testing data of one frame.

返回类型 dict

class mmtrack.datasets.**ImagenetVIDDataset**(**args*, ***kwargs*)

ImageNet VID dataset for video object detection.

load_annotations(*ann_file*)

Load annotations from COCO/COCOVID style annotation file.

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

返回 Annotation information from COCO/COCOVID api.

返回类型 list[dict]

load_image_anns(*ann_file*)

Load annotations from COCO style annotation file.

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

返回 Annotation information from COCO api.

返回类型 list[dict]

load_video_anns(*ann_file*)

Load annotations from COCOVID style annotation file.

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

返回 Annotation information from COCOVID api.

返回类型 list[dict]

class mmtrack.datasets.**LaSOTDataset**(**args*, ***kwargs*)

LaSOT dataset of single object tracking.

The dataset can both support training and testing mode.

get_visibility_from_video(*video_ind*)

Get the visible information of instance in a video.

load_data_infos(*split='test'*)

Load dataset information.

参数 **split** (*str, optional*) –Dataset split. Defaults to 'test'.

返回

The length of the list is the number of videos. The

inner dict is in the following format:

{ 'video_path': the video path 'ann_path': the annotation path 'start_frame_id': the starting frame number contained

in the image name

'end_frame_id': the ending frame number contained in the image name

'framename_template': the template of image name

}

返回类型 list[dict]

class mmtrack.datasets.**MOTChallengeDataset**(*visibility_thr=-1*, *interpolate_tracks_cfg=None*, *detection_file=None*, **args*, ***kwargs*)

Dataset for MOTChallenge.

参数

- **visibility_thr** (*float, optional*) –The minimum visibility for the objects during training. Default to -1.
- **interpolate_tracks_cfg** (*dict, optional*) –If not None, Interpolate tracks linearly to make tracks more complete. Defaults to None. - min_num_frames (int, optional): The minimum length of a track

  that will be interpolated. Defaults to 5.

  - **max_num_frames (int, optional): The maximum disconnected length** in a track. Defaults to 20.
- **detection_file** (*str, optional*) –The path of the public detection file. Default to None.

evaluate (*results*, *metric='track'*, *logger=None*, *resfile_path=None*, *bbox_iou_thr=0.5*, *track_iou_thr=0.5*)

Evaluation in MOT Challenge.

参数

- **results** (*list[list | tuple]*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated. Options are 'bbox', 'track'. Defaults to 'track'.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.
- **resfile_path** (*str, optional*) –Path to save the formatted results. Defaults to None.
- **bbox_iou_thr** (*float, optional*) –IoU threshold for detection evaluation. Defaults to 0.5.
- **track_iou_thr** (*float, optional*) –IoU threshold for tracking evaluation.. Defaults to 0.5.

返回 MOTChallenge style evaluation metric.

返回类型 dict[str, float]

format_bbox_results (*results*, *infos*, *resfile*)

Format detection results.

format_results (*results*, *resfile_path=None*, *metrics=['track']*)

Format the results to txts (standard format for MOT Challenge).

参数

- **results** (*dict(list[ndarray])*) –Testing results of the dataset.

- **resfile_path** (*str, optional*) –Path to save the formatted results. Defaults to None.
- **metrics** (*list[str], optional*) –The results of the specific metrics will be formatted.. Defaults to ['track'].

返回 (resfile_path, resfiles, names, tmp_dir), resfile_path is the path to save the formatted results, resfiles is a dict containing the filepaths, names is a list containing the name of the videos, tmp_dir is the temporal directory created for saving files.

返回类型 tuple

format_track_results(*results*, *infos*, *resfile*)

Format tracking results.

get_benchmark_and_eval_split()

Get benchmark and dataset split to evaluate.

Get benchmark from upeper/lower-case image prefix and the dataset split to evaluate.

返回 The first string denotes the type of dataset. The second string denotes the split of the dataset to eval.

返回类型 tuple(string)

get_dataset_cfg_for_hota(*gt_folder*, *tracker_folder*, *seqmap*)

Get default configs for trackeval.datasets.MotChallenge2DBox.

参数

- **gt_folder** (*str*) –the name of the GT folder
- **tracker_folder** (*str*) –the name of the tracker folder
- **seqmap** (*str*) –the file that contains the sequence of video names

返回 Dataset Configs for MotChallenge2DBox.

load_detections(*detection_file=None*)

Load public detections.

prepare_results(*img_info*)

Prepare results for image (e.g. the annotation information, ···).

class mmtrack.datasets.**OTB100Dataset**(**args*, ***kwargs*)

OTB100 dataset of single object tracking.

The dataset is only used to test.

get_bboxes_from_video(*video_ind*)

Get bboxes annotation about the instance in a video.

参数 **video_ind** (*int*) –video index

返回

in [N, 4] shape. The N is the bbox number and the bbox is in (x, y, w, h) format.

返回类型 ndarray

load_data_infos(*split='test'*)

Load dataset information.

参数 **split**(*str, optional*) –Dataset split. Defaults to 'test'.

返回

The length of the list is the number of videos. The

inner dict is in the following format:

{ 'video_path': the video path 'ann_path': the annotation path 'start_frame_id': the starting frame number contained

in the image name

'end_frame_id': the ending frame number contained in the image name

'framename_template': the template of image name 'init_skip_num': (optional) the number of skipped

frames when initializing tracker

}

返回类型 list[dict]

class mmtrack.datasets.**RandomSampleConcatDataset**(*dataset_cfgs*, *dataset_sampling_weights=None*)

A wrapper of concatenated dataset. Support randomly sampling one dataset from concatenated datasets and then getting samples from the sampled dataset.

参数

- **dataset_cfgs**(*list[dict]*) –The list contains all configs of concatenated datasets.
- **dataset_sampling_weights**(*list[float]*) –The list contains the sampling weights of each dataset.

class mmtrack.datasets.**ReIDDataset**(*pipeline*, *triplet_sampler=None*, **args*, ***kwargs*)

Dataset for ReID Dataset.

参数

- **pipeline**(*list*) –a list of dict, where each element represents a operation defined in *mmtrack.datasets.pipelines*
- **triplet_sampler**(*dict*) –The sampler for hard mining triplet loss.

evaluate(*results*, *metric='mAP'*, *metric_options=None*, *logger=None*)

Evaluate the ReID dataset.

参数

- **results** (*list*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated. Default value is *mAP*.
- **metric_options** –(dict, optional): Options for calculating metrics. Allowed keys are 'rank_list' and 'max_rank'. Defaults to None.
- **logger** (*logging.Logger | str, optional*) –Logger used for printing related information during evaluation. Defaults to None.

返回 evaluation results

返回类型 dict

load_annotations()

Load annotations from ImageNet style annotation file.

返回 Annotation information from ReID api.

返回类型 list[dict]

prepare_data(*idx*)

Prepare results for image (e.g. the annotation information, ···).

triplet_sampling(*pos_pid*, *num_ids=8*, *ins_per_id=4*)

Triplet sampler for hard mining triplet loss. First, for one pos_pid, random sample ins_per_id images with same person id.

Then, random sample num_ids - 1 negative ids. Finally, random sample ins_per_id images for each negative id.

参数

- **pos_pid** (*ndarray*) –The person id of the anchor.
- **num_ids** (*int*) –The number of person ids.
- **ins_per_id** (*int*) –The number of image for each person.

返回 Annotation information of num_ids X ins_per_id images.

返回类型 List

class mmtrack.datasets.**SOTCocoDataset**(*ann_file*, **args*, ***kwargs*)

Coco dataset of single object tracking.

The dataset only support training mode.

get_bboxes_from_video(*video_ind*)

Get bbox annotation about the instance in an image.

参数 **video_ind** (*int*) –video index. Each video_ind denotes an instance.

返回 in [1, 4] shape. The bbox is in (x, y, w, h) format.

返回类型 ndarray

get_img_infos_from_video(*video_ind*)

Get all frame paths in a video.

参数 **video_ind** (*int*) –video index. Each video_ind denotes an instance.

返回 all image paths

返回类型 list[str]

get_len_per_video(*video_ind*)

Get the number of frames in a video.

load_data_infos(*split='train'*)

Load dataset information. Each instance is viewed as a video.

参数 **split** (*str, optional*) –The split of dataset. Defaults to 'train'.

返回

The length of the list is the number of valid object annotations. The elemment in the list is annotation ID in coco API.

返回类型 list[int]

class mmtrack.datasets.**SOTImageNetVIDDataset**(*ann_file*, **args*, ***kwargs*)

ImageNet VID dataset of single object tracking.

The dataset only support training mode.

get_ann_infos_from_video(*video_ind*)

Get annotation information in a video. Note: We overload this function for speed up loading video information.

参数 **video_ind** (*int*) –video index. Each video_ind denotes an instance.

返回

{'bboxes': ndarray in (N, 4) shape, 'bboxes_isvalid': ndarray, 'visible':ndarray}. The bbox is in (x1, y1, x2, y2) format.

返回类型 dict

get_bboxes_from_video(*video_ind*)

Get bbox annotation about the instance in a video. Considering *get_bboxes_from_video* in *SOTBaseDataset* is not compatible with *SOTImageNetVIDDataset*, we oveload this function though it's not called by *self.get_ann_infos_from_video*.

参数 **video_ind** (*int*) –video index. Each video_ind denotes an instance.

返回 in [N, 4] shape. The bbox is in (x, y, w, h) format.

返回类型 ndarray

get_img_infos_from_video(*video_ind*)

Get image information in a video.

参数 **video_ind** (*int*) –video index

返回 {‘filename’: list[str], ‘frame_ids’:ndarray, ‘video_id’:int}

返回类型 dict

get_len_per_video(*video_ind*)

Get the number of frames in a video.

get_visibility_from_video(*video_ind*)

Get the visible information in a video.

Considering *get_visibility_from_video* in *SOTBaseDataset* is not compatible with *SOTImageNetVIDDataset*, we oveload this function though it’s not called by *self.get_ann_infos_from_video*.

load_data_infos(*split='train'*)

Load dataset information.

参数 **split** (*str, optional*) –The split of dataset. Defaults to ‘train’.

返回

The length of the list is the number of instances. The elemment in the list is instance ID in coco API.

返回类型 list[int]

class mmtrack.datasets.**SOTTestDataset**(**args*, ***kwargs*)

Dataset for the testing of single object tracking.

The dataset doesn’t support training mode.

evaluate(*results*, *metric=['track']*, *logger=None*)

Evaluation in OPE protocol.

参数

- **results** (*dict*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated. Options are ‘track’.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.

返回 OPE style evaluation metric (i.e. success, norm precision and precision).

返回类型 dict[str, float]

class mmtrack.datasets.**SOTTrainDataset**(**args*, ***kwargs*)

Dataset for the training of single object tracking.

The dataset doesn’t support testing mode.

get_snippet_of_instance(*idx*)

Get a snippet of an instance in a video.

参数 **idx** (*int*) –Index of data.

返回 (snippet, image_id, instance_id), snippet is a list containing the successive image ids where the instance appears, image_id is a random sampled image id from the snippet.

返回类型 tuple

load_video_anns(*ann_file*)

Load annotations from COCOVID style annotation file.

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

返回 Annotation information from COCOVID api.

返回类型 list[dict]

prepare_results(*img_id*, *instance_id*, *is_positive_pair*)

Get training data and annotations.

参数

- **img_id** (*int*) –The id of image.
- **instance_id** (*int*) –The id of instance.
- **is_positive_pair** (*bool*) –denoting positive or negative sample pair.

返回 The information of training image and annotation.

返回类型 dict

prepare_train_img(*idx*)

Get training data and annotations after pipeline.

参数 **idx** (*int*) –Index of data.

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

返回类型 dict

ref_img_sampling(*snippet*, *image_id*, *instance_id*, *frame_range=5*, *pos_prob=0.8*, *filter_key_img=False*, *return_key_img=True*, ***kwargs*)

Get a search image for an instance in an exemplar image.

If sampling a positive search image, the positive search image is randomly sampled from the exemplar image, where the sampled range is decided by *frame_range*. If sampling a negative search image, the negative search image and negative instance are randomly sampled from the entire dataset.

参数

- **snippet** (*list[int]*) –The successive image ids where the instance appears.
- **image_id** (*int*) –The id of exemplar image where the instance appears.
- **instance_id** (*int*) –The id of the instance.
- **frame_range** (*List(int) | int*) –The frame range of sampling a positive search image for the exemplar image. Default: 5.
- **pos_prob** (*float*) –The probability of sampling a positive search image. Default: 0.8.
- **filter_key_img** (*bool*) –If False, the exemplar image will be in the sampling candidates, otherwise, it is exclude. Default: False.
- **return_key_img** (*bool*) –If True, the *image_id* and *instance_id* are returned, otherwise, not returned. Default: True.

返回 (image_ids, instance_ids, is_positive_pair), image_ids is a list that must contain search image id and may contain *image_id*, instance_ids is a list that must contain search instance id and may contain *instance_id*, is_positive_pair is a bool denoting positive or negative sample pair.

返回类型 tuple

class mmtrack.datasets.**TaoDataset**(**args*, ***kwargs*)

Dataset for TAO.

evaluate(*results*, *metric=['bbox', 'track']*, *logger=None*, *resfile_path=None*)

Evaluation in COCO protocol and CLEAR MOT metric (e.g. MOTA, IDF1).

参数

- **results** (*dict*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated. Options are ‘bbox’, ‘segm’, ‘track’.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.
- **bbox_kwargs** (*dict*) –Configuration for COCO styple evaluation.
- **track_kwargs** (*dict*) –Configuration for CLEAR MOT evaluation.

返回 COCO style and CLEAR MOT evaluation metric.

返回类型 dict[str, float]

format_results(*results*, *resfile_path=None*)

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

参数

- **results** (*list[ndarray]*) –Testing results of the dataset.
- **resfile_path** (*str, optional*) –Path to save the formatted results. Defaults to None.

返回 (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 resfile_path is not specified.

返回类型 tuple

load_annotations(*ann_file*)

Load annotation from annotation file.

load_lvis_anns(*ann_file*)

Load annotation from COCO style annotation file.

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

返回 Annotation info from COCO api.

返回类型 list[dict]

load_tao_anns(*ann_file*)

Load annotation from COCOVID style annotation file.

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

返回 Annotation info from COCOVID api.

返回类型 list[dict]

class mmtrack.datasets.**TrackingNetDataset**(*chunks_list=['all']*, **args*, ***kwargs*)

TrackingNet dataset of single object tracking.

The dataset can both support training and testing mode.

format_results(*results*, *resfile_path=None*, *logger=None*)

Format the results to txts (standard format for TrackingNet Challenge).

参数

- **results** (*dict(list[ndarray])*) –Testing results of the dataset.
- **resfile_path** (*str*) –Path to save the formatted results. Defaults to None.
- **logger** (*logging.Logger | str | None, optional*) –defaults to None.

load_data_infos(*split='train'*)

Load dataset information.

参数 **split** (*str, optional*) –the split of dataset. Defaults to 'train'.

返回

the length of the list is the number of videos. The

inner dict is in the following format:

{ 'video_path': the video path 'ann_path': the annotation path 'start_frame_id': the starting frame ID number

contained in the image name

'**end_frame_id': the ending frame ID number contained in** the image name

'framename_template': the template of image name

}

返回类型 list[dict]

prepare_test_data(*video_ind*, *frame_ind*)

Get testing data of one frame. We parse one video, get one frame from it and pass the frame information to the pipeline.

参数

- **video_ind** (*int*) –video index
- **frame_ind** (*int*) –frame index

返回 testing data of one frame.

返回类型 dict

class mmtrack.datasets.**UAV123Dataset**(**args*, ***kwargs*)

UAV123 dataset of single object tracking.

The dataset is only used to test.

load_data_infos(*split='test'*)

Load dataset information.

参数 **split** (*str, optional*) –Dataset split. Defaults to 'test'.

返回

The length of the list is the number of videos. The

inner dict is in the following format:

{ 'video_path': the video path 'ann_path': the annotation path 'start_frame_id': the starting frame number contained

in the image name

'**end_frame_id': the ending frame number contained in** the image name

’framename_template’ : the template of image name

}

返回类型 list[dict]

class mmtrack.datasets.**VOTDataset**(*dataset_type='vot2018'*, **args*, ***kwargs*)

VOT dataset of single object tracking.

The dataset is only used to test.

evaluate(*results*, *metric=['track']*, *logger=None*, *interval=None*)

Evaluation in VOT protocol.

参数

- **results** (*dict*) –Testing results of the dataset. The tracking bboxes are in (tl_x, tl_y, br_x, br_y) format.
- **metric** (*str | list[str]*) –Metrics to be evaluated. Options are ‘track’.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.
- **interval** (*list*) –an specified interval in EAO curve used to calculate the EAO score. There are different settings in different VOT challenges.

返回

返回类型 dict[str, float]

get_ann_infos_from_video(*video_ind*)

Get bboxes annotation about the instance in a video.

参数 **video_ind** (*int*) –video index

返回

in [N, 8] shape. The N is the bbox number and the bbox is in (x1, y1, x2, y2, x3, y3, x4, y4) format.

返回类型 ndarray

load_data_infos(*split='test'*)

Load dataset information.

参数 **split** (*str, optional*) –Dataset split. Defaults to ‘test’.

返回

The length of the list is the number of videos. The

inner dict is in the following format:

{ ‘video_path’ : the video path ‘ann_path’ : the annotation path ‘start_frame_id’ : the starting frame number contained

in the image name

'**end_frame_id**': **the ending frame number contained in** the image name

'framename_template': the template of image name

}

返回类型 list[dict]

class mmtrack.datasets.**YouTubeVISDataset**(*dataset_version*, **args*, ***kwargs*)

YouTube VIS dataset for video instance segmentation.

convert_back_to_vis_format()

Convert the annotation back to the format of YouTube-VIS. The main difference between the two is the format of 'annotation'. Before modification, it is recorded in the unit of images, and after modification, it is recorded in the unit of instances.This operation is to make it easier to use the official eval API.

返回

A dict with 3 keys, categories, annotations and videos.

- categories (list[dict]): Each dict has 2 keys, id and name.
- videos (list[dict]): Each dict has 4 keys of video info, id, name, width and height.
- annotations (list[dict]): Each dict has 7 keys of video info, category_id, segmentations, bboxes, video_id, areas, id and iscrowd.

返回类型 dict

evaluate(*results*, *metric=['track_segm']*, *logger=None*)

Evaluation in COCO protocol.

参数

- **results** (*dict*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated. Options are 'track_segm'.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.

返回 COCO style evaluation metric.

返回类型 dict[str, float]

format_results(*results*, *resfile_path=None*, *metrics=['track_segm']*, *save_as_json=True*)

Format the results to a zip file (standard format for YouTube-VIS Challenge).

参数

- **results** (*dict(list[ndarray])*) –Testing results of the dataset.
- **resfile_path** (*str, optional*) –Path to save the formatted results. Defaults to None.
- **metrics** (*list[str], optional*) –The results of the specific metrics will be formatted. Defaults to [‘track_segm’].
- **save_as_json** (*bool, optional*) –Whether to save the json results file. Defaults to True.

返回 (resfiles, tmp_dir), resfiles is the path of the result json file, tmp_dir is the temporal directory created for saving files.

返回类型 tuple

mmtrack.datasets.**build_dataloader**(*dataset*, *samples_per_gpu*, *workers_per_gpu*, *num_gpus=1*, *samples_per_epoch=None*, *dist=True*, *shuffle=True*, *seed=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.

参数

- **dataset** (*Dataset*) –A PyTorch dataset.
- **samples_per_gpu** (*int*) –Number of training samples 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.
- **samples_per_epoch** (*int | None, Optional*) –The number of samples per epoch. If equal to -1, using all samples in the datasets per epoch. Otherwise, using the *samples_per_epoch* samples. Default: None.
- **dist** (*bool*) –Distributed training/test or not. Default: True.
- **shuffle** (*bool*) –Whether to shuffle the data at every epoch. Default: True.
- **seed** (*int, Optional*) –Seed to be used. Default: None.
- **persistent_workers** (*bool*) –If True, the data loader will not shutdown the worker processes after a dataset has been consumed once. This allows to maintain the workers *Dataset* instances alive. This argument is only valid when PyTorch>=1.7.0. Default: False.
- **kwargs** –any keyword argument to be used to initialize DataLoader

返回 A PyTorch dataloader.

返回类型 DataLoader

25.2 parsers

class mmtrack.datasets.parsers.**CocoVID**(**args: Any*, ***kwargs: Any*)

Inherit official COCO class in order to parse the annotations of bbox- related video tasks.

参数

- **annotation_file** (*str*) –location of annotation file. Defaults to None.
- **load_img_as_vid** (*bool*) –If True, convert image data to video data, which means each image is converted to a video. Defaults to False.

convert_img_to_vid(*dataset*)

Convert image data to video data.

createIndex()

Create index.

get_img_ids_from_ins_id(*insId*)

Get image ids from given instance id.

参数 **insId** (*int*) –The given instance id.

返回 Image ids of given instance id.

返回类型 list[int]

get_img_ids_from_vid(*vidId*)

Get image ids from given video id.

参数 **vidId** (*int*) –The given video id.

返回 Image ids of given video id.

返回类型 list[int]

get_ins_ids_from_vid(*vidId*)

Get instance ids from given video id.

参数 **vidId** (*int*) –The given video id.

返回 Instance ids of given video id.

返回类型 list[int]

get_vid_ids(*vidIds=[]*)

Get video ids that satisfy given filter conditions.

Default return all video ids.

参数 **vidIds** (*list[int]*) –The given video ids. Defaults to [].

返回 Video ids.

返回类型 list[int]

load_vids(*ids=[]*)

Get video information of given video ids.

Default return all videos information.

参数 **ids** (*list[int]*) –The given video ids. Defaults to [].

返回 List of video information.

返回类型 list[dict]

25.3 pipelines

class mmtrack.datasets.pipelines.**CheckPadMaskValidity**(*stride*)

Check the validity of data. Generally, it's used in such case: The image padding masks generated in the image preprocess need to be downsampled, and then passed into Transformer model, like DETR. The computation in the subsequent Transformer model must make sure that the values of downsampled mask are not all zeros.

参数 **stride** (*int*) –the max stride of feature map.

class mmtrack.datasets.pipelines.**ConcatSameTypeFrames**(*num_key_frames=1*)

Concat the frames of the same type. We divide all the frames into two types: 'key' frames and 'reference' frames.

The input list contains as least two dicts. We concat the first *num_key_frames* dicts to one dict, and the rest of dicts are concated to another dict.

In SOT field, 'key' denotes template image and 'reference' denotes search image.

参数 **num_key_frames** (*int, optional*) –the number of key frames. Defaults to 1.

concat_one_mode_results(*results*)

Concatenate the results of the same mode.

class mmtrack.datasets.pipelines.**ConcatVideoReferences**

Concat video references.

If the input list contains at least two dicts, concat the input list of dict to one dict from 2-nd dict of the input list.

Note: the 'ConcatVideoReferences' class will be deprecated in the future, please use 'ConcatSameTypeFrames' instead.

class mmtrack.datasets.pipelines.**LoadDetections**

Load public detections from MOT benchmark.

参数 **results** (*dict*) –Result dict from mmtrack.CocoVideoDataset.

class mmtrack.datasets.pipelines.**LoadMultiImagesFromFile**(**args*, ***kwargs*)

Load multi images from file.

Please refer to *mmdet.datasets.pipelines.loading.py:LoadImageFromFile* for detailed docstring.

class mmtrack.datasets.pipelines.**MatchInstances**(*skip_nomatch=True*)

Matching objects on a pair of images.

参数

- **skip_nomatch** (*bool, optional*) –Whether skip the pair of image
- **training when there are no matched objects. Default** (*during*) –
- **True.** (*to*) –

class mmtrack.datasets.pipelines.**PairSampling**(*frame_range=5*, *pos_prob=0.8*, *filter_template_img=False*)

Pair-style sampling. It’ s used in ‘SiameseRPN++

<https://arxiv.org/abs/1812.11703.>‘_.

参数

- **frame_range** (*List(int) | int*) –the sampling range of search frames in the same video for template frame. Defaults to 5.
- **pos_prob** (*float, optional*) –the probility of sampling positive sample pairs. Defaults to 0.8.
- **filter_template_img** (*bool, optional*) –if False, the template image will be in the sampling search candidates, otherwise, it is exclude. Defaults to False.

prepare_data(*video_info*, *sampled_inds*, *is_positive_pairs=False*)

Prepare sampled training data according to the sampled index.

参数

- **video_info** (*dict*) –the video information. It contains the keys: [‘bboxes’ ,’ bboxes_isvalid’ ,’ filename’ ,’ frame_ids’ , ‘video_id’ ,’ visible’].
- **sampled_inds** (*list[int]*) –the sampled frame indexes.
- **is_positive_pairs** (*bool, optional*) –whether it’ s the positive pairs. Defaults to False.

返回 contains the information of sampled data.

返回类型 List[dict]

class mmtrack.datasets.pipelines.**ReIDFormatBundle**(**args*, ***kwargs*)

ReID formatting bundle.

It first concatenates common fields, then simplifies the pipeline of formatting common fields, including “img” , and “gt_label” . These fields are formatted as follows.

- img: (1) transpose, (2) to tensor, (3) to DataContainer (stack=True)
- gt_labels: (1) to tensor, (2) to DataContainer

reid_format_bundle(*results*)

Transform and format gt_label fields in results.

参数 **results** (*dict*) –Result dict contains the data to convert.

返回 The result dict contains the data that is formatted with ReID bundle.

返回类型 dict

class mmtrack.datasets.pipelines.**SeqBboxJitter**(*scale_jitter_factor*, *center_jitter_factor*, *crop_size_factor*)

Bounding box jitter augmentation. The jittered bboxes are used for subsequent image cropping, like *SeqCropLikeStark*.

参数

- **scale_jitter_factor** (*list[int | float]*) –contains the factor of scale jitter.
- **center_jitter_factor** (*list[int | float]*) –contains the factor of center jitter.
- **crop_size_factor** (*list[int | float]*) –contains the ratio of crop size to bbox size.

class mmtrack.datasets.pipelines.**SeqBlurAug**(*prob=[0.0, 0.2]*)

Blur augmention for images.

参数 **prob** (*list[float]*) –The probability to perform blur augmention for each image. Defaults to [0.0, 0.2].

class mmtrack.datasets.pipelines.**SeqBrightnessAug**(*jitter_range=0*)

Brightness augmention for images.

参数 **jitter_range** (*float*) –The range of brightness jitter. Defaults to 0..

class mmtrack.datasets.pipelines.**SeqColorAug**(*prob=[1.0, 1.0]*, *rgb_var=[[- 0.55919361, 0.98062831, - 0.41940627], [1.72091413, 0.19879334, - 1.82968581], [4.64467907, 4.73710203, 4.88324118]]*)

Color augmention for images.

参数

- **prob** (*list[float]*) –The probability to perform color augmention for each image. Defaults to [1.0, 1.0].
- **rgb_var** (*list[list]*) –The values of color augmentaion. Defaults to [[-0.55919361, 0.98062831, -0.41940627], [1.72091413, 0.19879334, -1.82968581], [4.64467907, 4.73710203, 4.88324118]].

class mmtrack.datasets.pipelines.**SeqCropLikeSiamFC**(*context_amount=0.5*, *exemplar_size=127*, *crop_size=511*)

Crop images as SiamFC did.

The way of cropping an image is proposed in "Fully-Convolutional Siamese Networks for Object Tracking." SiamFC.

参数

- **context_amount** (*float*) –The context amount around a bounding box. Defaults to 0.5.
- **exemplar_size** (*int*) –Exemplar size. Defaults to 127.
- **crop_size** (*int*) –Crop size. Defaults to 511.

crop_like_SiamFC(*image*, *bbox*, *context_amount=0.5*, *exemplar_size=127*, *crop_size=511*)

Crop an image as SiamFC did.

参数

- **image** (*ndarray*) –of shape (H, W, 3).
- **bbox** (*ndarray*) –of shape (4,) in [x1, y1, x2, y2] format.
- **context_amount** (*float*) –The context amount around a bounding box. Defaults to 0.5.
- **exemplar_size** (*int*) –Exemplar size. Defaults to 127.
- **crop_size** (*int*) –Crop size. Defaults to 511.

返回 The cropped image of shape (crop_size, crop_size, 3).

返回类型 ndarray

generate_box(*image*, *gt_bbox*, *context_amount*, *exemplar_size*)

Generate box based on cropped image.

参数

- **image** (*ndarray*) –The cropped image of shape (self.crop_size, self.crop_size, 3).
- **gt_bbox** (*ndarray*) –of shape (4,) in [x1, y1, x2, y2] format.
- **context_amount** (*float*) –The context amount around a bounding box.
- **exemplar_size** (*int*) –Exemplar size. Defaults to 127.

返回 Generated box of shape (4,) in [x1, y1, x2, y2] format.

返回类型 ndarray

class mmtrack.datasets.pipelines.**SeqCropLikeStark**(*crop_size_factor*, *output_size*)

Crop images as Stark did.

The way of cropping an image is proposed in "Learning Spatio-Temporal Transformer for Visual Tracking." Stark.

参数

- **crop_size_factor** (*list[int | float]*) –contains the ratio of crop size to bbox size.
- **output_size** (*list[int | float]*) –contains the size of resized image (always square).

crop_like_stark(*img*, *bbox*, *crop_size_factor*, *output_size*)

Crop an image as Stark did.

参数

- **image** (*ndarray*) –of shape (H, W, 3).
- **bbox** (*ndarray*) –of shape (4,) in [x1, y1, x2, y2] format.
- **crop_size_factor** (*float*) –the ratio of crop size to bbox size
- **output_size** (*int*) –the size of resized image (always square).

返回

the cropped image of shape (crop_size, crop_size, 3).

resize_factor (float): the ratio of original image scale to cropped image scale.

pdding_mask (ndarray): the padding mask caused by cropping.

返回类型 img_crop_padded (ndarray)

generate_box(*bbox_gt*, *bbox_cropped*, *resize_factor*, *output_size*, *normalize=False*)

Transform the box coordinates from the original image coordinates to the coordinates of the cropped image.

参数

- **bbox_gt** (*ndarray*) –of shape (4,) in [x1, y1, x2, y2] format.
- **bbox_cropped** (*ndarray*) –of shape (4,) in [x1, y1, x2, y2] format.
- **resize_factor** (*float*) –the ratio of original image scale to cropped image scale.
- **output_size** (*float*) –the size of output image.
- **normalize** (*bool*) –whether to normalize the output box. Default to True.

返回 generated box of shape (4,) in [x1, y1, x2, y2] format.

返回类型 ndarray

class mmtrack.datasets.pipelines.**SeqDefaultFormatBundle**(*ref_prefix='ref'*)

Sequence Default formatting bundle.

It simplifies the pipeline of formatting common fields, including “img”, “img_metas”, “proposals”, “gt_bboxes”, “gt_instance_ids”, “gt_match_indices”, “gt_bboxes_ignore”, “gt_labels”, “gt_masks”, “gt_semantic_seg” and ‘padding_mask’. These fields are formatted as follows.

- img: (1) transpose, (2) to tensor, (3) to DataContainer (stack=True)
- img_metas: (1) to DataContainer (cpu_only=True)
- proposals: (1) to tensor, (2) to DataContainer
- gt_bboxes: (1) to tensor, (2) to DataContainer
- gt_instance_ids: (1) to tensor, (2) to DataContainer
- gt_match_indices: (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 DataContainer (cpu_only=True)
- gt_semantic_seg: (1) unsqueeze dim-0 (2) to tensor, (3) to DataContainer (stack=True)
- padding_mask: (1) to tensor, (2) to DataContainer

参数 **ref_prefix** (*str*) –The prefix of key added to the second dict of input list. Defaults to ‘ref’.

default_format_bundle(*results*)

Transform and format common fields in results.

参数 **results** (*dict*) –Result dict contains the data to convert.

返回 The result dict contains the data that is formatted with default bundle.

返回类型 dict

class mmtrack.datasets.pipelines.**SeqGrayAug**(*prob=0.0*)

Gray augmention for images.

参数 **prob** (*float*) –The probability to perform gray augmention. Defaults to 0..

class mmtrack.datasets.pipelines.**SeqLoadAnnotations**(*with_track=False*, **args*, ***kwargs*)

Sequence load annotations.

Please refer to *mmdet.datasets.pipelines.loading.py:LoadAnnotations* for detailed docstring.

参数 **with_track** (*bool*) –If True, load instance ids of bboxes.

class mmtrack.datasets.pipelines.**SeqNormalize**(**args*, ***kwargs*)

Normalize images.

Please refer to *mmdet.datasets.pipelines.transforms.py:Normalize* for detailed docstring.

class mmtrack.datasets.pipelines.**SeqPad**(**args*, ***kwargs*)

Pad images.

Please refer to *mmdet.datasets.pipelines.transforms.py:Pad* for detailed docstring.

class mmtrack.datasets.pipelines.**SeqPhotoMetricDistortion**(*share_params=True*, *brightness_delta=32*, *contrast_range=(0.5, 1.5)*, *saturation_range=(0.5, 1.5)*, *hue_delta=18*)

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

参数

- **brightness_delta** (*int*) –delta of brightness.
- **contrast_range** (*tuple*) –range of contrast.
- **saturation_range** (*tuple*) –range of saturation.
- **hue_delta** (*int*) –delta of hue.

get_params()

Generate parameters.

photo_metric_distortion(*results*, *params=None*)

Call function to perform photometric distortion on images.

参数

- **results** (*dict*) –Result dict from loading pipeline.
- **params** (*dict, optional*) –Pre-defined parameters. Default to None.

返回 Result dict with images distorted.

返回类型 dict

class mmtrack.datasets.pipelines.**SeqRandomCrop**(*crop_size*, *allow_negative_crop=False*, *share_params=False*, *bbox_clip_border=False*)

Sequentially random crop the images & bboxes & masks.

The absolute *crop_size* is sampled based on *crop_type* and *image_size*, then the cropped results are generated.

参数

- **crop_size** (*tuple*) –The relative ratio or absolute pixels of height and width.

- **allow_negative_crop** (*bool, optional*) –Whether to allow a crop that does not contain any bbox area. Default False.
- **share_params** (*bool, optional*) –Whether share the cropping parameters for the images.
- **bbox_clip_border** (*bool, optional*) –Whether clip the objects outside the border of the image. Defaults to True.

注解:

- **If the image is smaller than the absolute crop size, return the** original image.
- The keys for bboxes, labels and masks must be aligned. That is, *gt_bboxes* corresponds to *gt_labels* and *gt_masks*, and *gt_bboxes_ignore* corresponds to *gt_labels_ignore* and *gt_masks_ignore*.
- If the crop does not contain any gt-bbox region and *allow_negative_crop* is set to False, skip this image.

---

get_offsets(*img*)

Random generate the offsets for cropping.

random_crop(*results*, *offsets=None*)

Call function to randomly crop images, bounding boxes, masks, semantic segmentation maps.

参数

- **results** (*dict*) –Result dict from loading pipeline.
- **offsets** (*tuple, optional*) –Pre-defined offsets for cropping. Default to None.

返回 Randomly cropped results, 'img_shape' key in result dict is updated according to crop size.

返回类型 dict

class mmtrack.datasets.pipelines.**SeqRandomFlip**(*share_params*, **args*, ***kwargs*)

Randomly flip for images.

Please refer to *mmdet.datasets.pipelines.transforms.py:RandomFlip* for detailed docstring.

参数 **share_params** (*bool*) –If True, share the flip parameters for all images. Defaults to True.

class mmtrack.datasets.pipelines.**SeqResize**(*share_params=True*, **args*, ***kwargs*)

Resize images.

Please refer to *mmdet.datasets.pipelines.transforms.py:Resize* for detailed docstring.

参数 **share_params** (*bool*) –If True, share the resize parameters for all images. Defaults to True.

class mmtrack.datasets.pipelines.**SeqShiftScaleAug**(*target_size=[127, 255]*, *shift=[4, 64]*, *scale=[0.05, 0.18]*)

Shift and rescale images and bounding boxes.

参数

- **target_size** (*list[int]*) –list of int denoting exemplar size and search size, respectively. Defaults to [127, 255].
- **shift** (*list[int]*) –list of int denoting the max shift offset. Defaults to [4, 64].
- **scale** (*list[float]*) –list of float denoting the max rescale factor. Defaults to [0.05, 0.18].

class mmtrack.datasets.pipelines.**ToList**

Use list to warp each value of the input dict.

参数 **results** (*dict*) –Result dict contains the data to convert.

返回 Updated result dict contains the data to convert.

返回类型 dict

class mmtrack.datasets.pipelines.**TridentSampling**(*num_search_frames=1*, *num_template_frames=2*, *max_frame_range=[200]*, *cls_pos_prob=0.5*, *train_cls_head=False*, *min_num_frames=20*)

Multitemplate-style sampling in a trident manner. It's firstly used in STARK.

参数

- **num_search_frames** (*int, optional*) –the number of search frames
- **num_template_frames** (*int, optional*) –the number of template frames
- **max_frame_range** (*list[int], optional*) –the max frame range of sampling a positive search image for the template image. Its length is equal to the number of extra templates, i.e., *num_template_frames*-1. Default length is 1.
- **cls_pos_prob** (*float, optional*) –the probility of sampling positive samples in classification training.
- **train_cls_head** (*bool, optional*) –whether to train classification head.
- **min_num_frames** (*int, optional*) –the min number of frames to be sampled.

prepare_cls_data(*video_info*, *video_info_another*, *sampled_inds*)

Prepare the sampled classification training data according to the sampled index.

参数

- **video_info** (*dict*) –the video information. It contains the keys: ['bboxes','bboxes_isvalid','filename','frame_ids', 'video_id','visible'].
- **video_info_another** (*dict*) –the another video information. It's only used to get negative samples in classification train. It contains the keys: ['bboxes','bboxes_isvalid','filename', 'frame_ids','video_id','visible'].

- **sampled_inds** (*list[int]*) –the sampled frame indexes.

返回 contains the information of sampled data.

返回类型 List[dict]

prepare_data(*video_info*, *sampled_inds*, *with_label=False*)

Prepare sampled training data according to the sampled index.

参数

- **video_info** (*dict*) –the video information. It contains the keys: ['bboxes','bboxes_isvalid','filename','frame_ids', 'video_id','visible'].
- **sampled_inds** (*list[int]*) –the sampled frame indexes.
- **with_label** (*bool, optional*) –whether to recode labels in ann infos. Only set True in classification training. Defaults to False.

返回 contains the information of sampled data.

返回类型 List[dict]

random_sample_inds(*video_visibility*, *num_samples=1*, *frame_range=None*, *allow_invisible=False*, *force_invisible=False*)

Random sampling a specific number of samples from the specified frame range of the video. It also considers the visibility of each frame.

参数

- **video_visibility** (*ndarray*) –the visibility of each frame in the video.
- **num_samples** (*int, optional*) –the number of samples. Defaults to 1.
- **frame_range** (*list | None, optional*) –the frame range of sampling. Defaults to None.
- **allow_invisible** (*bool, optional*) –whether to allow to get invisible samples. Defaults to False.
- **force_invisible** (*bool, optional*) –whether to force to get invisible samples. Defaults to False.

返回 The sampled frame indexes.

返回类型 List

sampling_trident(*video_visibility*)

Sampling multiple template images and one search images in one video.

参数 **video_visibility** (*ndarray*) –the visibility of each frame in the video.

返回 the indexes of template and search images.

返回类型 List

class mmtrack.datasets.pipelines.**VideoCollect**(*keys*, *meta_keys=None*, *default_meta_keys=('filename', 'ori_filename', 'ori_shape', 'img_shape', 'pad_shape', 'scale_factor', 'flip', 'flip_direction', 'img_norm_cfg', 'frame_id', 'is_video_data')*)

Collect data from the loader relevant to the specific task.

参数

- **keys** (*Sequence[str]*) –Keys of results to be collected in `data`.
- **meta_keys** (*Sequence[str]*) –Meta keys to be converted to `mmcv.DataContainer` and collected in `data[img_metas]`. Defaults to None.
- **default_meta_keys** (*tuple*) –Default meta keys. Defaults to ('filename' , 'ori_filename' , 'ori_shape' , 'img_shape' , 'pad_shape' , 'scale_factor' , 'flip' , 'flip_direction' , 'img_norm_cfg' , 'frame_id' , 'is_video_data').

25.4 samplers

class mmtrack.datasets.samplers.**DistributedQuotaSampler**(*dataset*, *samples_per_epoch*, *num_replicas=None*, *rank=None*, *replacement=False*, *seed=0*)

Sampler that gets fixed number of samples per epoch.

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.

注解: Dataset is assumed to be of constant size.

参数

- **dataset** –Dataset used for sampling.
- **samples_per_epoch** (*int*) –The number of samples per epoch.
- **num_replicas** (*optional*) –Number of processes participating in distributed training.
- **rank** (*optional*) –Rank of the current process within num_replicas.
- **replacement** (*bool*) –samples are drawn with replacement if `True`, Default: False.
- **seed** (*int, optional*) –random seed used to shuffle the sampler if `shuffle=True`. This number should be identical across all processes in the distributed group. Default: 0.

class mmtrack.datasets.samplers.**DistributedVideoSampler**(*dataset*, *num_replicas=None*, *rank=None*, *shuffle=False*)

Put videos to multi gpus during testing.

参数

- **dataset** (*Dataset*) –Test dataset must have *data_infos* attribute. Each data_info in *data_infos* records information of one frame or one video (in SOT Dataset). If not SOT Dataset, each video must have one data_info that includes *data_info['frame_id'] == 0*.
- **num_replicas** (*int*) –The number of gpus. Defaults to None.
- **rank** (*int*) –Gpu rank id. Defaults to None.
- **shuffle** (*bool*) –If True, shuffle the dataset. Defaults to False.

class mmtrack.datasets.samplers.**SOTVideoSampler**(*dataset*)

Only used for sot testing on single gpu.

参数 **dataset** (*Dataset*) –Test dataset must have *num_frames_per_video* attribute. It records the frame number of each video.

CHAPTER 26

mmtrack.models

26.1 mot

class mmtrack.models.mot.**BaseMultiObjectTracker**(*init_cfg=None*)

Base class for multiple object tracking.

aug_test(*imgs*, *img_metas*, ***kwargs*)

Test function with test time augmentation.

forward(*img*, *img_metas*, *return_loss=True*, ***kwargs*)

Calls either *forward_train()* or *forward_test()* depending on whether `return_loss` is `True`.

Note this setting will change the expected inputs. When `return_loss=True`, img and img_meta are single-nested (i.e. Tensor and List[dict]), and when `resturn_loss=False`, img and img_meta should be double nested (i.e. List[Tensor], List[List[dict]]), with the outer list indicating test time augmentations.

forward_test(*imgs*, *img_metas*, ***kwargs*)

参数

- **imgs** (*List[Tensor]*) –the outer list indicates test-time augmentations and inner Tensor should have a shape NxCxHxW, which contains all images in the batch.
- **img_metas** (*List[List[dict]]*) –the outer list indicates test-time augs (multiscale, flip, etc.) and the inner list indicates images in a batch.

abstract forward_train(*imgs*, *img_metas*, ***kwargs*)

参数

- **img** (*list[Tensor]*) –List of tensors of shape (1, C, H, W). Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –List of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys, see `mmdet.datasets.pipelines.Collect`.
- **kwargs** (*keyword arguments*) –Specific to concrete implementation.

freeze_module(*module*)

Freeze module during training.

show_result(*img*, *result*, *score_thr=0.0*, *thickness=1*, *font_scale=0.5*, *show=False*, *out_file=None*, *wait_time=0*, *backend='cv2'*, ***kwargs*)

Visualize tracking results.

参数

- **img** (*str | ndarray*) –Filename of loaded image.
- **result** (*dict*) –Tracking result. - The value of key 'track_bboxes' is list with length num_classes, and each element in list is ndarray with shape(n, 6) in [id, tl_x, tl_y, br_x, br_y, score] format. - The value of key 'det_bboxes' is list with length num_classes, and each element in list is ndarray with shape(n, 5) in [tl_x, tl_y, br_x, br_y, score] format.
- **thickness** (*int, optional*) –Thickness of lines. Defaults to 1.
- **font_scale** (*float, optional*) –Font scales of texts. Defaults to 0.5.
- **show** (*bool, optional*) –Whether show the visualizations on the fly. Defaults to False.
- **out_file** (*str | None, optional*) –Output filename. Defaults to None.
- **backend** (*str, optional*) –Backend to draw the bounding boxes, options are *cv2* and *plt*. Defaults to 'cv2'.

返回 Visualized image.

返回类型 ndarray

abstract simple_test(*img*, *img_metas*, ***kwargs*)

Test function with a single scale.

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.

参数

- **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.

返回

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.

返回类型 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.

property with_detector

whether the framework has a detector.

Type bool

property with_motion

whether the framework has a motion model.

Type bool

property with_reid

whether the framework has a reid model.

Type bool

property with_track_head

whether the framework has a track_head.

Type bool

property with_tracker

whether the framework has a tracker.

Type bool

class mmtrack.models.mot.**ByteTrack**(*detector=None*, *tracker=None*, *motion=None*, *init_cfg=None*)

ByteTrack: Multi-Object Tracking by Associating Every Detection Box.

This multi object tracker is the implementation of ByteTrack.

参数

- **detector** (*dict*) –Configuration of detector. Defaults to None.
- **tracker** (*dict*) –Configuration of tracker. Defaults to None.
- **motion** (*dict*) –Configuration of motion. Defaults to None.
- **init_cfg** (*dict*) –Configuration of initialization. Defaults to None.

forward_train(**args*, ***kwargs*)

Forward function during training.

simple_test(*img*, *img_metas*, *rescale=False*, ***kwargs*)

Test without augmentations.

参数

- **img** (*Tensor*) –of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'.
- **rescale** (*bool, optional*) –If False, then returned bboxes and masks will fit the scale of img, otherwise, returned bboxes and masks will fit the scale of original image shape. Defaults to False.

返回 list(ndarray)]: The tracking results.

返回类型 dict[str

class mmtrack.models.mot.**DeepSORT**(*detector=None*, *reid=None*, *tracker=None*, *motion=None*, *pretrains=None*, *init_cfg=None*)

Simple online and realtime tracking with a deep association metric.

Details can be found at **'DeepSORT<https://arxiv.org/abs/1703.07402>'_**.

forward_train(**args*, ***kwargs*)

Forward function during training.

simple_test(*img*, *img_metas*, *rescale=False*, *public_bboxes=None*, ***kwargs*)

Test without augmentations.

参数

- **img** (*Tensor*) –of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.

- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'.
- **rescale** (*bool, optional*) –If False, then returned bboxes and masks will fit the scale of img, otherwise, returned bboxes and masks will fit the scale of original image shape. Defaults to False.
- **public_bboxes** (*list[Tensor], optional*) –Public bounding boxes from the benchmark. Defaults to None.

返回 list(ndarray)]: The tracking results.

返回类型 dict[str

class mmtrack.models.mot.**OCSORT**(*detector=None*, *tracker=None*, *motion=None*, *init_cfg=None*)

OCOSRT: Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking

This multi object tracker is the implementation of OC-SORT.

参数

- **detector** (*dict*) –Configuration of detector. Defaults to None.
- **tracker** (*dict*) –Configuration of tracker. Defaults to None.
- **motion** (*dict*) –Configuration of motion. Defaults to None.
- **init_cfg** (*dict*) –Configuration of initialization. Defaults to None.

forward_train(**args*, ***kwargs*)

Forward function during training.

simple_test(*img*, *img_metas*, *rescale=False*, ***kwargs*)

Test without augmentations.

参数

- **img** (*Tensor*) –of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'.
- **rescale** (*bool, optional*) –If False, then returned bboxes and masks will fit the scale of img, otherwise, returned bboxes and masks will fit the scale of original image shape. Defaults to False.

返回 list(ndarray)]: The tracking results.

返回类型 dict[str

class mmtrack.models.mot.**QDTrack**(*detector=None*, *track_head=None*, *tracker=None*, *freeze_detector=False*, **args*, ***kwargs*)

Quasi-Dense Similarity Learning for Multiple Object Tracking.

This multi object tracker is the implementation of QDTrack.

参数

- **detector** (*dict*) –Configuration of detector. Defaults to None.
- **track_head** (*dict*) –Configuration of track head. Defaults to None.
- **tracker** (*dict*) –Configuration of tracker. Defaults to None.
- **freeze_detector** (*bool*) –If True, freeze the detector weights. Defaults to False.

forward_train(*img*, *img_metas*, *gt_bboxes*, *gt_labels*, *gt_match_indices*, *ref_img*, *ref_img_metas*, *ref_gt_bboxes*, *ref_gt_labels*, *gt_bboxes_ignore=None*, *gt_masks=None*, *ref_gt_bboxes_ignore=None*, *ref_gt_masks=None*, ***kwargs*)

Forward function during training.

Args:

img (Tensor): of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.

img_metas (list[dict]): list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'.

gt_bboxes (list[Tensor]): Ground truth bboxes of the image, each item has a shape (num_gts, 4).

gt_labels (list[Tensor]): Ground truth labels of all images. each has a shape (num_gts,).

gt_match_indices (list(Tensor)): Mapping from gt_instance_ids to ref_gt_instance_ids of the same tracklet in a pair of images.

ref_img (Tensor): of shape (N, C, H, W) encoding input reference images. Typically these should be mean centered and std scaled.

ref_img_metas (list[dict]): list of reference image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'.

ref_gt_bboxes (list[Tensor]): Ground truth bboxes of the reference image, each item has a shape (num_gts, 4).

ref_gt_labels (list[Tensor]): Ground truth labels of all reference images, each has a shape (num_gts,).

gt_masks (list[Tensor]) [Masks for each bbox, has a shape] (num_gts, h , w).

gt_bboxes_ignore (list[Tensor], None): Ground truth bboxes to be ignored, each item has a shape (num_ignored_gts, 4).

ref_gt_bboxes_ignore (list[Tensor], None): Ground truth bboxes of reference images to be ignored, each item has a shape (num_ignored_gts, 4).

ref_gt_masks (list[Tensor]) [Masks for each reference bbox,] has a shape (num_gts, h , w).

返回 Tensor]: All losses.

返回类型 dict[str

simple_test(*img*, *img_metas*, *rescale=False*)

Test forward.

Args:

img (Tensor): of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.

img_metas (list[dict]): list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'.

rescale (bool): whether to rescale the bboxes.

返回 Tensor]: Track results.

返回类型 dict[str

class mmtrack.models.mot.**Tracktor**(*detector=None*, *reid=None*, *tracker=None*, *motion=None*, *pretrains=None*, *init_cfg=None*)

Tracking without bells and whistles.

Details can be found at **'Tracktor<https://arxiv.org/abs/1903.05625>'_**.

forward_train(**args*, ***kwargs*)

Forward function during training.

simple_test(*img*, *img_metas*, *rescale=False*, *public_bboxes=None*, ***kwargs*)

Test without augmentations.

参数

- **img** (*Tensor*) –of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'.

- **rescale** (*bool, optional*) –If False, then returned bboxes and masks will fit the scale of img, otherwise, returned bboxes and masks will fit the scale of original image shape. Defaults to False.
- **public_bboxes** (*list[Tensor], optional*) –Public bounding boxes from the benchmark. Defaults to None.

返回 list(ndarray)]: The tracking results.

返回类型 dict[str

property with_cmc

whether the framework has a camera model compensation model.

Type bool

property with_linear_motion

whether the framework has a linear motion model.

Type bool

26.2 sot

class mmtrack.models.sot.**SiamRPN**(*backbone*, *neck=None*, *head=None*, *pretrains=None*, *init_cfg=None*, *frozen_modules=None*, *train_cfg=None*, *test_cfg=None*)

SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks.

This single object tracker is the implementation of SiamRPN++.

forward_search(*x_img*)

Extract the features of search images.

参数 **x_img** (*Tensor*) –of shape (N, C, H, W) encoding input search images. Typically H and W equal to 255.

返回 Multi level feature map of search images.

返回类型 tuple(Tensor)

forward_template(*z_img*)

Extract the features of exemplar images.

参数 **z_img** (*Tensor*) –of shape (N, C, H, W) encoding input exemplar images. Typically H and W equal to 127.

返回 Multi level feature map of exemplar images.

返回类型 tuple(Tensor)

forward_train(*img*, *img_metas*, *gt_bboxes*, *search_img*, *search_img_metas*, *search_gt_bboxes*, *is_positive_pairs*, ***kwargs*)

参数

- **img** (*Tensor*) –of shape (N, C, H, W) encoding input exemplar images. Typically H and W equal to 127.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for each exemplar image with shape (1, 4) in [tl_x, tl_y, br_x, br_y] format.
- **search_img** (*Tensor*) –of shape (N, 1, C, H, W) encoding input search images. 1 denotes there is only one search image for each exemplar image. Typically H and W equal to 255.
- **search_img_metas** (*list[list[dict]]*) –The second list only has one element. The first list contains search image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **search_gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for each search image with shape (1, 5) in [0.0, tl_x, tl_y, br_x, br_y] format.
- **is_positive_pairs** (*list[bool]*) –list of bool denoting whether each exemplar image and corresponding search image is positive pair.

返回 a dictionary of loss components.

返回类型 dict[str, Tensor]

get_cropped_img(*img*, *center_xy*, *target_size*, *crop_size*, *avg_channel*)

Crop image.

Only used during testing.

This function mainly contains two steps: 1. Crop *img* based on center *center_xy* and size *crop_size*. If the cropped image is out of boundary of *img*, use *avg_channel* to pad. 2. Resize the cropped image to *target_size*.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding original input image.
- **center_xy** (*Tensor*) –of shape (2,) denoting the center point for cropping image.

- **target_size** (*int*) –The output size of cropped image.
- **crop_size** (*Tensor*) –The size for cropping image.
- **avg_channel** (*Tensor*) –of shape (3,) denoting the padding values.

返回 of shape (1, C, target_size, target_size) encoding the resized cropped image.

返回类型 Tensor

init(*img*, *bbox*)

Initialize the single object tracker in the first frame.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding original input image.
- **bbox** (*Tensor*) –The given instance bbox of first frame that need be tracked in the following frames. The shape of the box is (4,) with [cx, cy, w, h] format.

返回 z_feat is a tuple[Tensor] that contains the multi level feature maps of exemplar image, avg_channel is Tensor with shape (3,), and denotes the padding values.

返回类型 tuple(z_feat, avg_channel)

init_weights()

Initialize the weights of modules in single object tracker.

simple_test(*img*, *img_metas*, *gt_bboxes*, ***kwargs*)

Test without augmentation.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding input image.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **gt_bboxes** (*list[Tensor]*) –list of ground truth bboxes for each image with shape (1, 4) in [tl_x, tl_y, br_x, br_y] format or shape (1, 8) in [x1, y1, x2, y2, x3, y3, x4, y4].

返回 ndarray]: The tracking results.

返回类型 dict[str

simple_test_ope(*img*, *frame_id*, *gt_bboxes*)

Test using OPE test mode.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding input image.

- **frame_id** (*int*) –the id of current frame in the video.
- **gt_bboxes** (*list[Tensor]*) –list of ground truth bboxes for each image with shape (1, 4) in [tl_x, tl_y, br_x, br_y] format or shape (1, 8) in [x1, y1, x2, y2, x3, y3, x4, y4].

返回

in [tl_x, tl_y, br_x, br_y] format. best_score (Tensor): the tracking bbox confidence in range [0,1],

and the score of initial frame is -1.

返回类型 bbox_pred (Tensor)

simple_test_vot(*img*, *frame_id*, *gt_bboxes*, *img_metas=None*)

Test using VOT test mode.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding input image.
- **frame_id** (*int*) –the id of current frame in the video.
- **gt_bboxes** (*list[Tensor]*) –list of ground truth bboxes for each image with shape (1, 4) in [tl_x, tl_y, br_x, br_y] format or shape (1, 8) in [x1, y1, x2, y2, x3, y3, x4, y4].
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.

返回

in [tl_x, tl_y, br_x, br_y] format. best_score (Tensor): the tracking bbox confidence in range [0,1],

and the score of initial frame is -1.

返回类型 bbox_pred (Tensor)

track(*img*, *bbox*, *z_feat*, *avg_channel*)

Track the box *bbox* of previous frame to current frame *img*.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding original input image.
- **bbox** (*Tensor*) –The bbox in previous frame. The shape of the box is (4,) in [cx, cy, w, h] format.
- **z_feat** (*tuple[Tensor]*) –The multi level feature maps of exemplar image in the first frame.

- **avg_channel** (*Tensor*) –of shape (3,) denoting the padding values.

返回 best_score is a Tensor denoting the score of best_bbox, best_bbox is a Tensor of shape (4,) in [cx, cy, w, h] format, and denotes the best tracked bbox in current frame.

返回类型 tuple(best_score, best_bbox)

class mmtrack.models.sot.**Stark**(*backbone*, *neck=None*, *head=None*, *init_cfg=None*, *frozen_modules=None*, *train_cfg=None*, *test_cfg=None*)

STARK: Learning Spatio-Temporal Transformer for Visual Tracking.

This single object tracker is the implementation of STARk.

参数

- **backbone** (*dict*) –the configuration of backbone network.
- **neck** (*dict, optional*) –the configuration of neck network. Defaults to None.
- **head** (*dict, optional*) –the configuration of head network. Defaults to None.
- **init_cfg** (*dict, optional*) –the configuration of initialization. Defaults to None.
- **frozen_modules** (*str | list | tuple, optional*) –the names of frozen modules. Defaults to None.
- **train_cfg** (*dict, optional*) –the configuratioin of train. Defaults to None.
- **test_cfg** (*dict, optional*) –the configuration of test. Defaults to None.

extract_feat(*img*)

Extract the features of the input image.

参数 **img** (*Tensor*) –image of shape (N, C, H, W).

返回

the multi-level feature maps, and each of them is of shape (N, C, H // stride, W // stride).

返回类型 tuple(Tensor)

forward_train(*img*, *img_metas*, *search_img*, *search_img_metas*, *gt_bboxes*, *padding_mask*, *search_gt_bboxes*, *search_padding_mask*, *search_gt_labels=None*, ***kwargs*)

forward of training.

参数

- **img** (*Tensor*) –template images of shape (N, num_templates, C, H, W). Typically, there are 2 template images, and H and W are both equal to 128.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.

- **search_img** (*Tensor*) –of shape (N, 1, C, H, W) encoding input search images. 1 denotes there is only one search image for each template image. Typically H and W are both equal to 320.
- **search_img_metas** (*list[list[dict]]*) –The second list only has one element. The first list contains search image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for template images with shape (N, 4) in [tl_x, tl_y, br_x, br_y] format.
- **padding_mask** (*Tensor*) –padding mask of template images. It's of shape (N, num_templates, H, W). Typically, there are 2 padding masks of template images, and H and W are both equal to that of template images.
- **search_gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for search images with shape (N, 5) in [0., tl_x, tl_y, br_x, br_y] format.
- **search_padding_mask** (*Tensor*) –padding mask of search images. Its of shape (N, 1, H, W). There are 1 padding masks of search image, and H and W are both equal to that of search image.
- **search_gt_labels** (*list[Tensor], optional*) –Ground truth labels for search images with shape (N, 2).

返回 a dictionary of loss components.

返回类型 dict[str, Tensor]

get_cropped_img(*img*, *target_bbox*, *search_area_factor*, *output_size*)

Crop Image Only used during testing This function mainly contains two steps: 1. Crop *img* based on target_bbox and search_area_factor. If the cropped image/mask is out of boundary of *img*, use 0 to pad. 2. Resize the cropped image/mask to *output_size*.

参数

- **img** (*Tensor*) –of shape (1, C, H, W)
- **target_bbox** (*list | ndarray*) –in [cx, cy, w, h] format
- **search_area_factor** (*float*) –Ratio of crop size to target size
- **output_size** (*float*) –the size of output cropped image (always square).

返回

of shape (1, C, output_size, output_size) resize_factor (float): the ratio of original image scale to cropped

image scale.

pdding_mask (Tensor): the padding mask caused by cropping. It's of shape (1, output_size, output_size).

返回类型 img_crop_padded (Tensor)

init(*img*, *bbox*)

Initialize the single object tracker in the first frame.

参数

- **img** (*Tensor*) –input image of shape (1, C, H, W).
- **bbox** (*list | Tensor*) –in [cx, cy, w, h] format.

init_weights()

Initialize the weights of modules in single object tracker.

mapping_bbox_back(*pred_bboxes*, *prev_bbox*, *resize_factor*)

Mapping the *prediction bboxes* from resized cropped image to original image. The coordinate origins of them are both the top left corner.

参数

- **pred_bboxes** (*Tensor*) –the predicted bbox of shape (B, Nq, 4), in [tl_x, tl_y, br_x, br_y] format. The coordinates are based in the resized cropped image.
- **prev_bbox** (*Tensor*) –the previous bbox of shape (B, 4), in [cx, cy, w, h] format. The coordinates are based in the original image.
- **resize_factor** (*float*) –the ratio of original image scale to cropped image scale.

返回 in [tl_x, tl_y, br_x, br_y] format.

返回类型 (Tensor)

simple_test(*img*, *img_metas*, *gt_bboxes*, ***kwargs*)

Test without augmentation.

参数

- **img** (*Tensor*) –input image of shape (1, C, H, W).
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **gt_bboxes** (*list[Tensor]*) –list of ground truth bboxes for each image with shape (1, 4) in [tl_x, tl_y, br_x, br_y] format.

返回 ndarray): the tracking results.

返回类型 dict(str

track(*img*, *bbox*)

Track the box *bbox* of previous frame to current frame *img*.

参数

- **img** (*Tensor*) –of shape (1, C, H, W).
- **bbox** (*list | Tensor*) –The bbox in previous frame. The shape of the bbox is (4,) in [x, y, w, h] format.

Returns:

update_template(*img*, *bbox*, *conf_score*)

Update the dymanic templates.

参数

- **img** (*Tensor*) –of shape (1, C, H, W).
- **bbox** (*list | ndarray*) –in [cx, cy, w, h] format.
- **conf_score** (*float*) –the confidence score of the predicted bbox.

26.3 vid

class mmtrack.models.vid.**BaseVideoDetector**(*init_cfg*)

Base class for video object detector.

参数 **init_cfg** (*dict or list[dict], optional*) –Initialization config dict.

aug_test(*imgs*, *img_metas*, ***kwargs*)

Test function with test time augmentation.

forward(*img*, *img_metas*, *ref_img=None*, *ref_img_metas=None*, *return_loss=True*, ***kwargs*)

Calls either *forward_train()* or *forward_test()* depending on whether `return_loss` is `True`.

Note this setting will change the expected inputs. When `return_loss=True`, img and img_meta are single-nested (i.e. Tensor and List[dict]), and when `resturn_loss=False`, img and img_meta should be double nested (i.e. List[Tensor], List[List[dict]]), with the outer list indicating test time augmentations.

forward_test(*imgs*, *img_metas*, *ref_img=None*, *ref_img_metas=None*, ***kwargs*)

参数

- **imgs** (*List[Tensor]*) –the outer list indicates test-time augmentations and inner Tensor should have a shape NxCxHxW, which contains all images in the batch.

- **img_metas** (*List[List[dict]]*) –the outer list indicates test-time augs (multiscale, flip, etc.) and the inner list indicates images in a batch.
- **ref_img** (*list[Tensor] | None*) –The list only contains one Tensor of shape (1, N, C, H, W) encoding input reference images. Typically these should be mean centered and std scaled. N denotes the number for reference images. There may be no reference images in some cases.
- **ref_img_metas** (*list[list[list[dict]]] | None*) –The first and second list only has one element. The third list contains image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*. There may be no reference images in some cases.

abstract forward_train(*imgs*, *img_metas*, *ref_img=None*, *ref_img_metas=None*, ***kwargs*)

参数

- **img** (*Tensor*) –of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **ref_img** (*Tensor*) –of shape (N, R, C, H, W) encoding input images. Typically these should be mean centered and std scaled. R denotes there is #R reference images for each input image.
- **ref_img_metas** (*list[list[dict]]*) –The first list only has one element. The second list contains reference image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.

freeze_module(*module*)

Freeze module during training.

show_result(*img*, *result*, *score_thr=0.3*, *bbox_color='green'*, *text_color='green'*, *thickness=1*, *font_scale=0.5*, *win_name=''*, *show=False*, *wait_time=0*, *out_file=None*)

Draw *result* over *img*.

参数

- **img** (*str or Tensor*) –The image to be displayed.

- **result** (*dict*) –The results to draw over *img* det_bboxes or (det_bboxes, det_masks). The value of key 'det_bboxes' is list with length num_classes, and each element in list is ndarray with shape(n, 5) in [tl_x, tl_y, br_x, br_y, score] format.
- **score_thr** (*float, optional*) –Minimum score of bboxes to be shown. Default: 0.3.
- **bbox_color** (str or tuple or `Color`) –Color of bbox lines.
- **text_color** (str or tuple or `Color`) –Color of texts.
- **thickness** (*int*) –Thickness of lines.
- **font_scale** (*float*) –Font scales of texts.
- **win_name** (*str*) –The window name.
- **wait_time** (*int*) –Value of waitKey param. Default: 0.
- **show** (*bool*) –Whether to show the image. Default: False.
- **out_file** (*str or None*) –The filename to write the image. Default: None.

返回 Only if not *show* or *out_file*

返回类型 img (Tensor)

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.

参数

- **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.

返回

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.

返回类型 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.

property with_aggregator

whether the framework has a aggregator

Type bool

property with_detector

whether the framework has a detector

Type bool

property with_motion

whether the framework has a motion model

Type bool

class mmtrack.models.vid.**DFF**(*detector*, *motion*, *pretrains=None*, *init_cfg=None*, *frozen_modules=None*, *train_cfg=None*, *test_cfg=None*)

Deep Feature Flow for Video Recognition.

This video object detector is the implementation of DFF.

aug_test(*imgs*, *img_metas*, ***kwargs*)

Test function with test time augmentation.

extract_feats(*img*, *img_metas*)

Extract features for *img* during testing.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding input image. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.

返回 Multi level feature maps of *img*.

返回类型 list[Tensor]

forward_train(*img*, *img_metas*, *gt_bboxes*, *gt_labels*, *ref_img*, *ref_img_metas*, *ref_gt_bboxes*, *ref_gt_labels*, *gt_instance_ids=None*, *gt_bboxes_ignore=None*, *gt_masks=None*, *proposals=None*, *ref_gt_instance_ids=None*, *ref_gt_bboxes_ignore=None*, *ref_gt_masks=None*, *ref_proposals=None*, ***kwargs*)

参数

- **img** (*Tensor*) –of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for each image with shape (num_gts, 4) in [tl_x, tl_y, br_x, br_y] format.
- **gt_labels** (*list[Tensor]*) –class indices corresponding to each box.
- **ref_img** (*Tensor*) –of shape (N, 1, C, H, W) encoding input images. Typically these should be mean centered and std scaled. 1 denotes there is only one reference image for each input image.
- **ref_img_metas** (*list[list[dict]]*) –The first list only has one element. The second list contains reference image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **ref_gt_bboxes** (*list[Tensor]*) –The list only has one Tensor. The Tensor contains ground truth bboxes for each reference image with shape (num_all_ref_gts, 5) in [ref_img_id, tl_x, tl_y, br_x, br_y] format. The ref_img_id start from 0, and denotes the id of reference image for each key image.
- **ref_gt_labels** (*list[Tensor]*) –The list only has one Tensor. The Tensor contains class indices corresponding to each reference box with shape (num_all_ref_gts, 2) in [ref_img_id, class_indice].
- **gt_instance_ids** (*None | list[Tensor]*) –specify the instance id for each ground truth bbox.
- **gt_bboxes_ignore** (*None | list[Tensor]*) –specify which bounding boxes can be ignored when computing the loss.
- **gt_masks** (*None | Tensor*) –true segmentation masks for each box used if the architecture supports a segmentation task.
- **proposals** (*None | Tensor*) –override rpn proposals with custom proposals. Use when *with_rpn* is False.
- **ref_gt_instance_ids** (*None | list[Tensor]*) –specify the instance id for each ground truth bboxes of reference images.

- **ref_gt_bboxes_ignore** (*None | list[Tensor]*) –specify which bounding boxes of reference images can be ignored when computing the loss.
- **ref_gt_masks** (*None | Tensor*) –True segmentation masks for each box of reference image used if the architecture supports a segmentation task.
- **ref_proposals** (*None | Tensor*) –override rpn proposals with custom proposals of reference images. Use when *with_rpn* is False.

返回 a dictionary of loss components

返回类型 dict[str, Tensor]

simple_test(*img*, *img_metas*, *ref_img=None*, *ref_img_metas=None*, *proposals=None*, *rescale=False*)

Test without augmentation.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding input image. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **ref_img** (*None*) –Not used in DFF. Only for unifying API interface.
- **ref_img_metas** (*None*) –Not used in DFF. Only for unifying API interface.
- **proposals** (*None | Tensor*) –Override rpn proposals with custom proposals. Use when *with_rpn* is False. Defaults to None.
- **rescale** (*bool*) –If False, then returned bboxes and masks will fit the scale of img, otherwise, returned bboxes and masks will fit the scale of original image shape. Defaults to False.

返回 list(ndarray)]: The detection results.

返回类型 dict[str

class mmtrack.models.vid.**FGFA**(*detector*, *motion*, *aggregator*, *pretrains=None*, *init_cfg=None*, *frozen_modules=None*, *train_cfg=None*, *test_cfg=None*)

Flow-Guided Feature Aggregation for Video Object Detection.

This video object detector is the implementation of FGFA.

aug_test(*imgs*, *img_metas*, ***kwargs*)

Test function with test time augmentation.

extract_feats(*img*, *img_metas*, *ref_img*, *ref_img_metas*)

Extract features for *img* during testing.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding input image. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **ref_img** (*Tensor | None*) –of shape (1, N, C, H, W) encoding input reference images. Typically these should be mean centered and std scaled. N denotes the number of reference images. There may be no reference images in some cases.
- **ref_img_metas** (*list[list[dict]] | None*) –The first list only has one element. The second list contains image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*. There may be no reference images in some cases.

返回 Multi level feature maps of *img*.

返回类型 list[Tensor]

forward_train(*img*, *img_metas*, *gt_bboxes*, *gt_labels*, *ref_img*, *ref_img_metas*, *ref_gt_bboxes*, *ref_gt_labels*, *gt_instance_ids=None*, *gt_bboxes_ignore=None*, *gt_masks=None*, *proposals=None*, *ref_gt_instance_ids=None*, *ref_gt_bboxes_ignore=None*, *ref_gt_masks=None*, *ref_proposals=None*, ***kwargs*)

参数

- **img** (*Tensor*) –of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for each image with shape (num_gts, 4) in [tl_x, tl_y, br_x, br_y] format.
- **gt_labels** (*list[Tensor]*) –class indices corresponding to each box.
- **ref_img** (*Tensor*) –of shape (N, 2, C, H, W) encoding input images. Typically these should be mean centered and std scaled. 2 denotes there is two reference images for each input image.

- **ref_img_metas** (*list[list[dict]]*) –The first list only has one element. The second list contains reference image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **ref_gt_bboxes** (*list[Tensor]*) –The list only has one Tensor. The Tensor contains ground truth bboxes for each reference image with shape (num_all_ref_gts, 5) in [ref_img_id, tl_x, tl_y, br_x, br_y] format. The ref_img_id start from 0, and denotes the id of reference image for each key image.
- **ref_gt_labels** (*list[Tensor]*) –The list only has one Tensor. The Tensor contains class indices corresponding to each reference box with shape (num_all_ref_gts, 2) in [ref_img_id, class_indice].
- **gt_instance_ids** (*None | list[Tensor]*) –specify the instance id for each ground truth bbox.
- **gt_bboxes_ignore** (*None | list[Tensor]*) –specify which bounding boxes can be ignored when computing the loss.
- **gt_masks** (*None | Tensor*) –true segmentation masks for each box used if the architecture supports a segmentation task.
- **proposals** (*None | Tensor*) –override rpn proposals with custom proposals. Use when *with_rpn* is False.
- **ref_gt_instance_ids** (*None | list[Tensor]*) –specify the instance id for each ground truth bboxes of reference images.
- **ref_gt_bboxes_ignore** (*None | list[Tensor]*) –specify which bounding boxes of reference images can be ignored when computing the loss.
- **ref_gt_masks** (*None | Tensor*) –True segmentation masks for each box of reference image used if the architecture supports a segmentation task.
- **ref_proposals** (*None | Tensor*) –override rpn proposals with custom proposals of reference images. Use when *with_rpn* is False.

返回 a dictionary of loss components

返回类型 dict[str, Tensor]

simple_test(*img*, *img_metas*, *ref_img=None*, *ref_img_metas=None*, *proposals=None*, *rescale=False*)

Test without augmentation.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding input image. Typically these should be mean centered and std scaled.

- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **ref_img** (*list[Tensor] | None*) –The list only contains one Tensor of shape (1, N, C, H, W) encoding input reference images. Typically these should be mean centered and std scaled. N denotes the number for reference images. There may be no reference images in some cases.
- **ref_img_metas** (*list[list[list[dict]]] | None*) –The first and second list only has one element. The third list contains image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*. There may be no reference images in some cases.
- **proposals** (*None | Tensor*) –Override rpn proposals with custom proposals. Use when *with_rpn* is False. Defaults to None.
- **rescale** (*bool*) –If False, then returned bboxes and masks will fit the scale of img, otherwise, returned bboxes and masks will fit the scale of original image shape. Defaults to False.

返回 list(ndarray)]: The detection results.

返回类型 dict[str

class mmtrack.models.vid.**SELSA**(*detector*, *pretrains=None*, *init_cfg=None*, *frozen_modules=None*, *train_cfg=None*, *test_cfg=None*)

Sequence Level Semantics Aggregation for Video Object Detection.

This video object detector is the implementation of SELSA.

aug_test(*imgs*, *img_metas*, ***kwargs*)

Test function with test time augmentation.

extract_feats(*img*, *img_metas*, *ref_img*, *ref_img_metas*)

Extract features for *img* during testing.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding input image. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.

- **ref_img** (*Tensor | None*) –of shape (1, N, C, H, W) encoding input reference images. Typically these should be mean centered and std scaled. N denotes the number of reference images. There may be no reference images in some cases.
- **ref_img_metas** (*list[list[dict]] | None*) –The first list only has one element. The second list contains image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*. There may be no reference images in some cases.

返回

x is the multi level feature maps of *img*, ref_x is the multi level feature maps of *ref_img*.

返回类型 tuple(x, img_metas, ref_x, ref_img_metas)

forward_train(*img*, *img_metas*, *gt_bboxes*, *gt_labels*, *ref_img*, *ref_img_metas*, *ref_gt_bboxes*, *ref_gt_labels*, *gt_instance_ids=None*, *gt_bboxes_ignore=None*, *gt_masks=None*, *proposals=None*, *ref_gt_instance_ids=None*, *ref_gt_bboxes_ignore=None*, *ref_gt_masks=None*, *ref_proposals=None*, ***kwargs*)

参数

- **img** (*Tensor*) –of shape (N, C, H, W) encoding input images. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for each image with shape (num_gts, 4) in [tl_x, tl_y, br_x, br_y] format.
- **gt_labels** (*list[Tensor]*) –class indices corresponding to each box.
- **ref_img** (*Tensor*) –of shape (N, 2, C, H, W) encoding input images. Typically these should be mean centered and std scaled. 2 denotes there is two reference images for each input image.
- **ref_img_metas** (*list[list[dict]]*) –The first list only has one element. The second list contains reference image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **ref_gt_bboxes** (*list[Tensor]*) –The list only has one Tensor. The Tensor contains ground truth bboxes for each reference image with shape (num_all_ref_gts,

5) in [ref_img_id, tl_x, tl_y, br_x, br_y] format. The ref_img_id start from 0, and denotes the id of reference image for each key image.

- **ref_gt_labels** (*list[Tensor]*) –The list only has one Tensor. The Tensor contains class indices corresponding to each reference box with shape (num_all_ref_gts, 2) in [ref_img_id, class_indice].
- **gt_instance_ids** (*None | list[Tensor]*) –specify the instance id for each ground truth bbox.
- **gt_bboxes_ignore** (*None | list[Tensor]*) –specify which bounding boxes can be ignored when computing the loss.
- **gt_masks** (*None | Tensor*) –true segmentation masks for each box used if the architecture supports a segmentation task.
- **proposals** (*None | Tensor*) –override rpn proposals with custom proposals. Use when *with_rpn* is False.
- **ref_gt_instance_ids** (*None | list[Tensor]*) –specify the instance id for each ground truth bboxes of reference images.
- **ref_gt_bboxes_ignore** (*None | list[Tensor]*) –specify which bounding boxes of reference images can be ignored when computing the loss.
- **ref_gt_masks** (*None | Tensor*) –True segmentation masks for each box of reference image used if the architecture supports a segmentation task.
- **ref_proposals** (*None | Tensor*) –override rpn proposals with custom proposals of reference images. Use when *with_rpn* is False.

返回 a dictionary of loss components

返回类型 dict[str, Tensor]

simple_test(*img*, *img_metas*, *ref_img=None*, *ref_img_metas=None*, *proposals=None*, *ref_proposals=None*, *rescale=False*)

Test without augmentation.

参数

- **img** (*Tensor*) –of shape (1, C, H, W) encoding input image. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **ref_img** (*list[Tensor] | None*) –The list only contains one Tensor of shape (1, N, C, H, W) encoding input reference images. Typically these should be

mean centered and std scaled. N denotes the number for reference images. There may be no reference images in some cases.

- **ref_img_metas** (*list[list[list[dict]]] | None*) –The first and second list only has one element. The third list contains image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*. There may be no reference images in some cases.
- **proposals** (*None | Tensor*) –Override rpn proposals with custom proposals. Use when *with_rpn* is False. Defaults to None.
- **rescale** (*bool*) –If False, then returned bboxes and masks will fit the scale of img, otherwise, returned bboxes and masks will fit the scale of original image shape. Defaults to False.

返回 list(ndarray)]: The detection results.

返回类型 dict[str

26.4 aggregators

class mmtrack.models.aggregators.**EmbedAggregator**(*num_convs=1*, *channels=256*, *kernel_size=3*, *norm_cfg=None*, *act_cfg={'type': 'ReLU'}*, *init_cfg=None*)

Embedding convs to aggregate multi feature maps.

This module is proposed in "Flow-Guided Feature Aggregation for Video Object Detection". FGFA.

参数

- **num_convs** (*int*) –Number of embedding convs.
- **channels** (*int*) –Channels of embedding convs. Defaults to 256.
- **kernel_size** (*int*) –Kernel size of embedding convs, Defaults to 3.
- **norm_cfg** (*dict*) –Configuration of normlization method after each conv. Defaults to None.
- **act_cfg** (*dict*) –Configuration of activation method after each conv. Defaults to dict(type='ReLU').
- **init_cfg** (*dict or list[dict], optional*) –Initialization config dict. Defaults to None.

forward(*x*, *ref_x*)

Aggregate reference feature maps *ref_x*.

The aggregation mainly contains two steps: 1. Computing the cos similarity between *x* and *ref_x*. 2. Use the normlized (i.e. softmax) cos similarity to weightedly sum *ref_x*.

参数

- **x** (*Tensor*) –of shape [1, C, H, W]
- **ref_x** (*Tensor*) –of shape [N, C, H, W]. N is the number of reference feature maps.

返回 The aggregated feature map with shape [1, C, H, W].

返回类型 Tensor

class mmtrack.models.aggregators.**SelsaAggregator**(*in_channels*, *num_attention_blocks=16*, *init_cfg=None*)

Selsa aggregator module.

This module is proposed in "Sequence Level Semantics Aggregation for Video Object Detection". SELSA.

参数

- **in_channels** (*int*) –The number of channels of the features of proposal.
- **num_attention_blocks** (*int*) –The number of attention blocks used in selsa aggregator module. Defaults to 16.
- **init_cfg** (*dict or list[dict], optional*) –Initialization config dict. Defaults to None.

forward(*x*, *ref_x*)

Aggregate the features *ref_x* of reference proposals.

The aggregation mainly contains two steps: 1. Use multi-head attention to computing the weight between *x* and *ref_x*. 2. Use the normlized (i.e. softmax) weight to weightedly sum *ref_x*.

参数

- **x** (*Tensor*) –of shape [N, C]. N is the number of key frame proposals.
- **ref_x** (*Tensor*) –of shape [M, C]. M is the number of reference frame proposals.

返回 The aggregated features of key frame proposals with shape [N, C].

返回类型 Tensor

26.5 backbones

class mmtrack.models.backbones.**SOTResNet**(*depth*, *unfreeze_backbone=True*, ***kwargs*)

ResNet backbone for SOT.

The main difference between ResNet in torch and the SOTResNet is the padding and dilation in the convs of SOTResNet. Please refer to SiamRPN++ for detailed analysis.

参数 **depth** (*int*) –Depth of resnet, from {50, }.

make_res_layer(***kwargs*)

Pack all blocks in a stage into a ResLayer.

26.6 losses

class mmtrack.models.losses.**L2Loss**(*neg_pos_ub=- 1*, *pos_margin=- 1*, *neg_margin=- 1*, *hard_mining=False*, *reduction='mean'*, *loss_weight=1.0*)

L2 loss.

参数

- **reduction** (*str, optional*) –The method to reduce the loss. Options are "none", "mean" and "sum".
- **loss_weight** (*float, optional*) –The weight of loss.

forward(*pred*, *target*, *weight=None*, *avg_factor=None*, *reduction_override=None*)

Forward function.

参数

- **pred** (*torch.Tensor*) –The prediction.
- **target** (*torch.Tensor*) –The learning target of the prediction.
- **weight** (*torch.Tensor, optional*) –The weight of loss for each prediction. Defaults to None.
- **avg_factor** (*int, optional*) –Average factor that is used to average the loss. Defaults to None.
- **reduction_override** (*str, optional*) –The reduction method used to override the original reduction method of the loss. Defaults to None.

static random_choice(*gallery*, *num*)

Random select some elements from the gallery.

It seems that Pytorch's implementation is slower than numpy so we use numpy to randperm the indices.

update_weight(*pred*, *target*, *weight*, *avg_factor*)

Update the weight according to targets.

class mmtrack.models.losses.**MultiPosCrossEntropyLoss**(*reduction='mean'*, *loss_weight=1.0*)

multi-positive targets cross entropy loss.

forward(*cls_score*, *label*, *weight=None*, *avg_factor=None*, *reduction_override=None*, ***kwargs*)

Forward function.

参数

- **cls_score** (*torch.Tensor*) –The classification score.
- **label** (*torch.Tensor*) –The assigned label of the prediction.
- **weight** (*torch.Tensor*) –The element-wise weight.
- **avg_factor** (*float*) –Average factor when computing the mean of losses.
- **reduction** (*str*) –Same as built-in losses of PyTorch.

返回 Calculated loss

返回类型 torch.Tensor

multi_pos_cross_entropy(*pred*, *label*, *weight=None*, *reduction='mean'*, *avg_factor=None*)

参数

- **pred** (*torch.Tensor*) –The prediction.
- **label** (*torch.Tensor*) –The assigned label of the prediction.
- **weight** (*torch.Tensor*) –The element-wise weight.
- **reduction** (*str*) –Same as built-in losses of PyTorch.
- **avg_factor** (*float*) –Average factor when computing the mean of losses.

返回 Calculated loss

返回类型 torch.Tensor

class mmtrack.models.losses.**TripletLoss**(*margin=0.3*, *loss_weight=1.0*, *hard_mining=True*)

Triplet loss with hard positive/negative mining.

Reference:

Hermans et al. In Defense of the Triplet Loss for Person Re-Identification. arXiv:1703.07737.

Imported from ‘<https://github.com/KaiyangZhou/deep-person-reid/blob/ master/torchreid/losses/hard_mine_triplet_loss.py>‘_.

参数

- **margin** (*float, optional*) –Margin for triplet loss. Default to 0.3.
- **loss_weight** (*float, optional*) –Weight of the loss. Default to 1.0.

forward(*inputs*, *targets*, ***kwargs*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解： 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.

---

hard_mining_triplet_loss_forward(*inputs*, *targets*)

参数

- **inputs** (*torch.Tensor*) –feature matrix with shape (batch_size, feat_dim).
- **targets** (*torch.LongTensor*) –ground truth labels with shape (num_classes).

26.7 motion

class mmtrack.models.motion.**CameraMotionCompensation**(*warp_mode='cv2.MOTION_EUCLIDEAN'*, *num_iters=50*, *stop_eps=0.001*)

Camera motion compensation.

参数

- **warp_mode** (*str*) –Warp mode in opencv.
- **num_iters** (*int*) –Number of the iterations.
- **stop_eps** (*float*) –Terminate threshold.

get_warp_matrix(*img*, *ref_img*)

Calculate warping matrix between two images.

track(*img*, *ref_img*, *tracks*, *num_samples*, *frame_id*)

Tracking forward.

warp_bboxes(*bboxes*, *warp_matrix*)

Warp bounding boxes according to the warping matrix.

class mmtrack.models.motion.**FlowNetSimple**(*img_scale_factor*, *out_indices=[2, 3, 4, 5, 6]*, *flow_scale_factor=5.0*, *flow_img_norm_std=[255.0, 255.0, 255.0]*, *flow_img_norm_mean=[0.411, 0.432, 0.45]*, *init_cfg=None*)

The simple version of FlowNet.

This FlowNetSimple is the implementation of FlowNetSimple.

参数

- **img_scale_factor** (*float*) –Used to upsample/downsample the image.
- **out_indices** (*list*) –The indices of outputting feature maps after each group of conv layers. Defaults to [2, 3, 4, 5, 6].
- **flow_scale_factor** (*float*) –Used to enlarge the values of flow. Defaults to 5.0.
- **flow_img_norm_std** (*list*) –Used to scale the values of image. Defaults to [255.0, 255.0, 255.0].
- **flow_img_norm_mean** (*list*) –Used to center the values of image. Defaults to [0.411, 0.432, 0.450].
- **init_cfg** (*dict or list[dict], optional*) –Initialization config dict. Defaults to None.

crop_like(*input*, *target*)

Crop *input* as the size of *target*.

forward(*imgs*, *img_metas*)

Compute the flow of images pairs.

参数

- **imgs** (*Tensor*) –of shape (N, 6, H, W) encoding input images pairs. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.

返回 of shape (N, 2, H, W) encoding flow of images pairs.

返回类型 Tensor

prepare_imgs(*imgs*, *img_metas*)

Preprocess images pairs for computing flow.

参数

- **imgs** (*Tensor*) –of shape (N, 6, H, W) encoding input images pairs. Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –list of image information dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.

返回 of shape (N, 6, H, W) encoding the input images pairs for FlowNetSimple.

返回类型 Tensor

class mmtrack.models.motion.**KalmanFilter**(*center_only=False*)

A simple Kalman filter for tracking bounding boxes in image space.

The implementation is referred to https://github.com/nwojke/deep_sort.

gating_distance(*mean*, *covariance*, *measurements*, *only_position=False*)

Compute gating distance between state distribution and measurements.

A suitable distance threshold can be obtained from *chi2inv95*. If *only_position* is False, the chi-square distribution has 4 degrees of freedom, otherwise 2.

参数

- **mean** (*ndarray*) –Mean vector over the state distribution (8 dimensional).
- **covariance** (*ndarray*) –Covariance of the state distribution (8x8 dimensional).
- **measurements** (*ndarray*) –An Nx4 dimensional matrix of N measurements, each in format (x, y, a, h) where (x, y) is the bounding box center position, a the aspect ratio, and h the height.
- **only_position** (*bool, optional*) –If True, distance computation is done with respect to the bounding box center position only. Defaults to False.

返回 Returns an array of length N, where the i-th element contains the squared Mahalanobis distance between (mean, covariance) and *measurements[i]*.

返回类型 ndarray

initiate(*measurement*)

Create track from unassociated measurement.

参数

- **measurement** (*ndarray*) –Bounding box coordinates (x, y, a, h) with
- **position** (*center*) –

返回

Returns the mean vector (8 dimensional) and covariance matrix (8x8 dimensional) of the new track. Unobserved velocities are initialized to 0 mean.

返回类型 (ndarray, ndarray)

predict(*mean*, *covariance*)

Run Kalman filter prediction step.

参数

- **mean** (*ndarray*) –The 8 dimensional mean vector of the object state at the previous time step.

- **covariance** (*ndarray*) –The 8x8 dimensional covariance matrix of the object state at the previous time step.

返回

Returns the mean vector and covariance matrix of the predicted state. Unobserved velocities are initialized to 0 mean.

返回类型 (ndarray, ndarray)

project (*mean*, *covariance*)

Project state distribution to measurement space.

参数

- **mean** (*ndarray*) –The state' s mean vector (8 dimensional array).
- **covariance** (*ndarray*) –The state' s covariance matrix (8x8 dimensional).

返回 Returns the projected mean and covariance matrix of the given state estimate.

返回类型 (ndarray, ndarray)

track (*tracks*, *bboxes*)

Track forward.

参数

- **(dict[int** (*tracks*) –dict]): Track buffer.
- **bboxes** (*Tensor*) –Detected bounding boxes.

返回 dict], Tensor): Updated tracks and bboxes.

返回类型 (dict[int

update (*mean*, *covariance*, *measurement*)

Run Kalman filter correction step.

参数

- **mean** (*ndarray*) –The predicted state' s mean vector (8 dimensional).
- **covariance** (*ndarray*) –The state' s covariance matrix (8x8 dimensional).
- **measurement** (*ndarray*) –The 4 dimensional measurement vector (x, y, a, h), where (x, y) is the center position, a the aspect ratio, and h the height of the bounding box.

返回 Returns the measurement-corrected state distribution.

返回类型 (ndarray, ndarray)

class mmtrack.models.motion.**LinearMotion** (*num_samples=2*, *center_motion=False*)

Linear motion while tracking.

参数

- **num_samples** (*int, optional*) –Number of samples to calculate the velocity. Default to 2.
- **center_motion** (*bool, optional*) –Whether use center location or bounding box location to estimate the velocity. Default to False.

center(*bbox*)

Get the center of the box.

get_velocity(*bboxes*, *num_samples=None*)

Get velocities of the input objects.

step(*bboxes*, *velocity=None*)

Step forward with the velocity.

track(*tracks*, *frame_id*)

Tracking forward.

26.8 reid

class mmtrack.models.reid.**BaseReID**(*backbone*, *neck=None*, *head=None*, *pretrained=None*, *train_cfg=None*, *init_cfg=None*)

Base class for re-identification.

forward_train(*img*, *gt_label*, ***kwargs*)

"Training forward function.

simple_test(*img*, ***kwargs*)

Test without augmentation.

class mmtrack.models.reid.**FcModule**(*in_channels*, *out_channels*, *norm_cfg=None*, *act_cfg={'type': 'ReLU'}*, *inplace=True*, *init_cfg={'layer': 'Linear', 'type': 'Kaiming'}*)

Fully-connected layer module.

参数

- **in_channels** (*int*) –Input channels.
- **out_channels** (*int*) –Ourput channels.
- **norm_cfg** (*dict, optional*) –Configuration of normlization method after fc. Defaults to None.
- **act_cfg** (*dict, optional*) –Configuration of activation method after fc. Defaults to dict(type='ReLU').
- **inplace** (*bool, optional*) –Whether inplace the activatation module.
- **init_cfg** (*dict or list[dict], optional*) –Initialization config dict. Defaults to dict(type='Kaiming', layer='Linear').

forward(*x*, *activate=True*, *norm=True*)

Model forward.

property norm

Normalization.

class mmtrack.models.reid.**GlobalAveragePooling**(*kernel_size=None*, *stride=None*)

Global Average Pooling neck.

Note that we use *view* to remove extra channel after pooling. We do not use *squeeze* as it will also remove the batch dimension when the tensor has a batch dimension of size 1, which can lead to unexpected errors.

class mmtrack.models.reid.**LinearReIDHead**(*num_fcs*, *in_channels*, *fc_channels*, *out_channels*, *norm_cfg=None*, *act_cfg=None*, *num_classes=None*, *loss=None*, *loss_pairwise=None*, *topk=(1)*, *init_cfg={'bias': 0, 'layer': 'Linear', 'mean': 0, 'std': 0.01, 'type': 'Normal'}*)

Linear head for re-identification.

参数

- **num_fcs** (*int*) –Number of fcs.
- **in_channels** (*int*) –Number of channels in the input.
- **fc_channels** (*int*) –Number of channels in the fcs.
- **out_channels** (*int*) –Number of channels in the output.
- **norm_cfg** (*dict, optional*) –Configuration of normlization method after fc. Defaults to None.
- **act_cfg** (*dict, optional*) –Configuration of activation method after fc. Defaults to None.
- **num_classes** (*int, optional*) –Number of the identities. Default to None.
- **loss** (*dict, optional*) –Cross entropy loss to train the re-identificaiton module.
- **loss_pairwise** (*dict, optional*) –Triplet loss to train the re-identificaiton module.
- **topk** (*int, optional*) –Calculate topk accuracy. Default to False.
- **init_cfg** (*dict or list[dict], optional*) –Initialization config dict. Defaults to dict(type='Normal',layer='Linear', mean=0, std=0.01, bias=0).

forward_train(*x*)

Model forward.

loss(*gt_label*, *feats*, *cls_score=None*)

Compute losses.

26.9 roi_heads

class mmtrack.models.roi_heads.**SelsaBBoxHead**(*aggregator*, **args*, ***kwargs*)

Selsa bbox head.

This module is proposed in "Sequence Level Semantics Aggregation for Video Object Detection". SELSA.

参数 **aggregator** (*dict*) –Configuration of aggregator.

forward(*x*, *ref_x*)

Computing the *cls_score* and *bbox_pred* of the features *x* of key frame proposals.

参数

- **x** (*Tensor*) –of shape [N, C, H, W]. N is the number of key frame proposals.
- **ref_x** (*Tensor*) –of shape [M, C, H, W]. M is the number of reference frame proposals.

返回 The predicted score of classes and the predicted regression offsets.

返回类型 tuple(cls_score, bbox_pred)

class mmtrack.models.roi_heads.**SelsaRoIHead**(*bbox_roi_extractor=None*, *bbox_head=None*, *mask_roi_extractor=None*, *mask_head=None*, *shared_head=None*, *train_cfg=None*, *test_cfg=None*, *pretrained=None*, *init_cfg=None*)

selsa roi head.

forward_train(*x*, *ref_x*, *img_metas*, *proposal_list*, *ref_proposal_list*, *gt_bboxes*, *gt_labels*, *gt_bboxes_ignore=None*, *gt_masks=None*)

参数

- **x** (*list[Tensor]*) –list of multi-level img features.
- **ref_x** (*list[Tensor]*) –list of multi-level ref_img features.
- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmdet/datasets/pipelines/formatting.py:Collect*.
- **proposal_list** (*list[Tensors]*) –list of region proposals.
- **ref_proposal_list** (*list[Tensors]*) –list of region proposals from ref_imgs.
- **gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for each image with shape (num_gts, 4) in [tl_x, tl_y, br_x, br_y] format.
- **gt_labels** (*list[Tensor]*) –class indices corresponding to each box

- **gt_bboxes_ignore** (*None | list[Tensor]*) –specify which bounding boxes can be ignored when computing the loss.
- **gt_masks** (*None | Tensor*) –true segmentation masks for each box used if the architecture supports a segmentation task.

返回 a dictionary of loss components

返回类型 dict[str, Tensor]

simple_test(*x*, *ref_x*, *proposals_list*, *ref_proposals_list*, *img_metas*, *proposals=None*, *rescale=False*)
Test without augmentation.

simple_test_bboxes(*x*, *ref_x*, *proposals*, *ref_proposals*, *img_metas*, *rcnn_test_cfg*, *rescale=False*)
Test only det bboxes without augmentation.

class mmtrack.models.roi_heads.**SingleRoIExtractor**(*roi_layer*, *out_channels*, *featmap_strides*, *finest_scale=56*, *init_cfg=None*)

Extract RoI features from a single level feature map.

This Class is the same as *SingleRoIExtractor* from *mmdet.models.roi_heads.roi_extractors* except for using ***kwargs* to accept external arguments.

forward(*feats*, *rois*, *roi_scale_factor=None*, ***kwargs*)
Forward function.

class mmtrack.models.roi_heads.**TemporalRoIAlign**(*num_most_similar_points=2*, *num_temporal_attention_blocks=4*, **args*, ***kwargs*)

Temporal RoI Align module.

This module is proposed in "Temporal ROI Align for Video Object Recognition". TRoI Align.

参数

- **num_most_similar_points** (*int*) –Denotes the number of the most similar points in the Most Similar RoI Align. Defaults to 2.
- **num_temporal_attention_blocks** (*int*) –Denotes the number of temporal attention blocks in the Temporal Attentional Feature Aggregation. If the value isn't greater than 0, the averaging operation will be adopted to aggregate the RoI features with the Most Similar RoI features. Defaults to 4.

forward(*feats*, *rois*, *roi_scale_factor=None*, *ref_feats=None*)
Forward function.

most_similar_roi_align(*roi_feats*, *ref_feats*)
Extract the Most Similar RoI features from reference feature maps *ref_feats* based on RoI features *roi_feats*.

The extraction mainly contains three steps: 1. Compute cos similarity maps between *roi_feats* and *ref_feats*. 2. Pick the top K points based on the similarity maps. 3. Project these top K points into reference feature maps *ref_feats*.

参数

- **roi_feats** (*Tensor*) –of shape [roi_n, C, roi_h, roi_w]. roi_n, roi_h and roi_w denote the number of key frame proposals, the height of RoI features and the width of RoI features, respectively.
- **ref_feats** (*Tensor*) –of shape [img_n, C, img_h, img_w]. img_n, img_h and img_w denote the number of reference frames, the height of reference frame feature maps and the width of reference frame feature maps, respectively.

返回

The extracted Most Similar RoI features from reference feature maps with shape [img_n, roi_n, C, roi_h, roi_w].

返回类型 Tensor

temporal_attentional_feature_aggregation(*x*, *ref_x*)

Aggregate the RoI features *x* with the Most Similar RoI features *ref_x*.

The aggregation mainly contains three steps: 1. Pass through a tiny embed network. 2. Use multi-head attention to computing the weight between *x* and *ref_x*. 3. Use the normlized (i.e. softmax) weight to weightedly sum *x* and *ref_x*.

参数

- **x** (*Tensor*) –of shape [1, roi_n, C, roi_h, roi_w]. roi_n, roi_h and roi_w denote the number of key frame proposals, the height of RoI features and the width of RoI features, respectively.
- **ref_x** (*Tensor*) –of shape [img_n, roi_n, C, roi_h, roi_w]. img_n is the number of reference images.

返回

The aggregated Temporal RoI features of key frame proposals with shape [roi_n, C, roi_h, roi_w].

返回类型 Tensor

26.10 track_heads

class mmtrack.models.track_heads.**CornerPredictorHead**(*inplanes*, *channel*, *feat_size=20*, *stride=16*)

Corner Predictor head.

参数

- **inplanes** (*int*) –input channel
- **channel** (*int*) –the output channel of the first conv block

- **feat_size** (*int*) –the size of feature map
- **stride** (*int*) –the stride of feature map from the backbone

forward(*x*)

Forward pass with input x.

参数 **x** (*Tensor*) –of shape (bs, C, H, W).

返回 bbox of shape (bs, 4) in (tl_x, tl_y, br_x, br_y) format.

返回类型 (Tensor)

get_score_map(*x*)

Score map branch.

参数 **x** (*Tensor*) –of shape (bs, C, H, W).

返回

of shape (bs, 1, H, W). The score map of top left corner of tracking bbox.

score_map_br (Tensor): of shape (bs, 1, H, W). The score map of bottom right corner of tracking bbox.

返回类型 score_map_tl (Tensor)

soft_argmax(*score_map*)

Get soft-argmax coordinate for the given score map.

参数 **score_map** (*self.feat_size, self.feat_size*) –the last score map in bbox_head branch

返回

of shape (bs, 1). The values are in range [0, self.feat_size * self.stride]

exp_y (Tensor): of shape (bs, 1). The values are in range [0, self.feat_size * self.stride]

返回类型 exp_x (Tensor)

class mmtrack.models.track_heads.**CorrelationHead**(*in_channels*, *mid_channels*, *out_channels*, *kernel_size=3*, *norm_cfg={'type': 'BN'}*, *act_cfg={'type': 'ReLU'}*, *init_cfg=None*, ***kwargs*)

Correlation head module.

This module is proposed in "SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks. SiamRPN++.

参数

- **in_channels** (*int*) –Input channels.
- **mid_channels** (*int*) –Middle channels.

- **out_channels** (*int*) –Output channels.
- **kernel_size** (*int*) –Kernel size of convs. Defaults to 3.
- **norm_cfg** (*dict*) –Configuration of normlization method after each conv. Defaults to dict(type='BN').
- **act_cfg** (*dict*) –Configuration of activation method after each conv. Defaults to dict(type='ReLU').
- **init_cfg** (*dict or list[dict], optional*) –Initialization config dict. Defaults to None.

forward(*kernel*, *search*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: 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.

---

class mmtrack.models.track_heads.**QuasiDenseEmbedHead**(*embed_channels=256*, *softmax_temp=- 1*, *loss_track={'loss_weight': 0.25, 'type': 'MultiPosCrossEntropyLoss'}*, *loss_track_aux={'hard_mining': True, 'loss_weight': 1.0, 'margin': 0.3, 'sample_ratio': 3, 'type': 'L2Loss'}*, *init_cfg={'bias': 0, 'distribution': 'uniform', 'layer': 'Linear', 'override': {'bias': 0, 'mean': 0, 'name': 'fc_embed', 'std': 0.01, 'type': 'Normal'}, 'type': 'Xavier'}*, **args*, ***kwargs*)

The quasi-dense roi embed head.

参数

- **embed_channels** (*int*) –The input channel of embed features. Defaults to 256.
- **softmax_temp** (*int*) –Softmax temperature. Defaults to -1.
- **loss_track** (*dict*) –The loss function for tracking. Defaults to MultiPosCrossEntropyLoss.
- **loss_track_aux** (*dict*) –The auxiliary loss function for tracking. Defaults to L2Loss.

forward(*x*)

Forward the input *x*.

get_targets(*gt_match_indices*, *key_sampling_results*, *ref_sampling_results*)

Calculate the track targets and track weights for all samples in a batch according to the sampling_results.

参数

- **(List[obj** (*ref_sampling_results*) –SamplingResults]): Assign results of all images in a batch after sampling.
- **(List[obj** –SamplingResults]): Assign results of all reference images in a batch after sampling.
- **gt_match_indices** (*list(Tensor)*) –Mapping from gt_instance_ids to ref_gt_instance_ids of the same tracklet in a pair of images.

返回

Association results. Containing the following list of Tensors:

- **track_targets (list[Tensor]): The mapping instance ids from** all positive proposals in the key image to all proposals in the reference image, each tensor in list has shape (len(key_pos_bboxes), len(ref_bboxes)).
- **track_weights (list[Tensor]): Loss weights for all positive** proposals in a batch, each tensor in list has shape (len(key_pos_bboxes),).

返回类型 Tuple[list[Tensor]]

loss(*dists*, *cos_dists*, *targets*, *weights*)

Calculate the track loss and the auxiliary track loss.

参数

- **dists** (*list[Tensor]*) –Dot-product dists between key_embeds and ref_embeds.
- **cos_dists** (*list[Tensor]*) –Cosine dists between key_embeds and ref_embeds.
- **targets**(*list[Tensor]*) –The mapping instance ids from all positive proposals in the key image to all proposals in the reference image, each tensor in list has shape (len(key_pos_bboxes), len(ref_bboxes)).
- **weights** (*list[Tensor]*) –Loss weights for all positive proposals in a batch, each tensor in list has shape (len(key_pos_bboxes),).

返回

Tensor]: Calculation results. Containing the following list of Tensors:

- loss_track (Tensor): Results of loss_track function.

- loss_track_aux (Tensor): Results of loss_track_aux function.

返回类型 Dict [str

match(*key_embeds*, *ref_embeds*, *key_sampling_results*, *ref_sampling_results*)

Calculate the dist matrixes for loss measurement.

参数

- **key_embeds** (*Tensor*) –Embeds of positive bboxes in sampling results of key image.
- **ref_embeds** (*Tensor*) –Embeds of all bboxes in sampling results of the reference image.
- **(List[obj** (*ref_sampling_results*) –SamplingResults]): Assign results of all images in a batch after sampling.
- **(List[obj** –SamplingResults]): Assign results of all reference images in a batch after sampling.

返回

Calculation results. Containing the following list of Tensors:

- **dists (list[Tensor]): Dot-product dists between** key_embeds and ref_embeds, each tensor in list has shape (len(key_pos_bboxes), len(ref_bboxes)).
- **cos_dists (list[Tensor]): Cosine dists between** key_embeds and ref_embeds, each tensor in list has shape (len(key_pos_bboxes), len(ref_bboxes)).

返回类型 Tuple[list[Tensor]]

class mmtrack.models.track_heads.**QuasiDenseTrackHead**(**args*, ***kwargs*)

The quasi-dense track head.

extract_bbox_feats(*x*, *bboxes*)

Extract roi features.

forward_train(*x*, *img_metas*, *proposal_list*, *gt_bboxes*, *gt_labels*, *gt_match_indices*, *ref_x*, *ref_img_metas*, *ref_proposals*, *ref_gt_bboxes*, *ref_gt_labels*, *gt_bboxes_ignore=None*, *gt_masks=None*, *ref_gt_bboxes_ignore=None*, *ref_gt_mask=None*, **args*, ***kwargs*)

Forward function during training.

Args: x (list[Tensor]): list of multi-level image features. img_metas (list[dict]): list of image info dict where each dict

has: ‘img_shape’, ‘scale_factor’, ‘flip’, and may also contain ‘filename’, ‘ori_shape’, ‘pad_shape’, and ‘img_norm_cfg’.

proposal_list (list[Tensors]): list of region proposals. gt_bboxes (list[Tensor]): Ground truth bboxes of the image,

each item has a shape (num_gts, 4).

gt_labels (list[Tensor]): Ground truth labels of all images. each has a shape (num_gts,).

gt_match_indices (list(Tensor)): Mapping from gt_instance_ids to ref_gt_instance_ids of the same tracklet in a pair of images.

ref_x (list[Tensor]): list of multi-level ref_img features. ref_img_metas (list[dict]): list of reference image info dict where

each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'.

ref_proposal_list (list[Tensors]): list of ref_img region proposals.

ref_gt_bboxes (list[Tensor]): Ground truth bboxes of the reference image, each item has a shape (num_gts, 4).

ref_gt_labels (list[Tensor]): Ground truth labels of all reference images, each has a shape (num_gts,).

gt_bboxes_ignore (list[Tensor], None): Ground truth bboxes to be ignored, each item has a shape (num_ignored_gts, 4).

gt_masks (list[Tensor]) [Masks for each bbox, has a shape] (num_gts, h , w).

ref_gt_bboxes_ignore (list[Tensor], None): Ground truth bboxes of reference images to be ignored, each item has a shape (num_ignored_gts, 4).

ref_gt_masks (list[Tensor]) [Masks for each reference bbox,] has a shape (num_gts, h , w).

返回 Tensor]: Track losses.

返回类型 dict[str

class mmtrack.models.track_heads.**RoIEmbedHead**(*num_convs=0*, *num_fcs=0*, *roi_feat_size=7*, *in_channels=256*, *conv_out_channels=256*, *with_avg_pool=False*, *fc_out_channels=1024*, *conv_cfg=None*, *norm_cfg=None*, *loss_match={'loss_weight': 1.0, 'type': 'CrossEntropyLoss', 'use_sigmoid': False}*, *init_cfg=None*, ***kwargs*)

The roi embed head.

This module is used in multi-object tracking methods, such as MaskTrack R-CNN.

参数

- **num_convs** (*int*) –The number of convoluational layers to embed roi features. Defaults to 0.
- **num_fcs** (*int*) –The number of fully connection layers to embed roi features. Defaults to 0.
- **roi_feat_size** (*int | tuple(int)*) –The spatial size of roi features. Defaults to 7.
- **in_channels** (*int*) –The input channel of roi features. Defaults to 256.
- **conv_out_channels** (*int*) –The output channel of roi features after forwarding convoluational layers. Defaults to 256.
- **with_avg_pool** (*bool*) –Whether use average pooling before passing roi features into fully connection layers. Defaults to False.
- **fc_out_channels** (*int*) –The output channel of roi features after forwarding fully connection layers. Defaults to 1024.
- **conv_cfg** (*dict*) –Config dict for convolution layer. Defaults to None, which means using conv2d.
- **norm_cfg** (*dict*) –Config dict for normalization layer. Defaults to None.
- **loss_match** (*dict*) –The loss function. Defaults to dict(type=' CrossEntropyLoss' , use_sigmoid=False, loss_weight=1.0)
- **init_cfg** (*dict*) –Configuration of initialization. Defaults to None.

forward(*x*, *ref_x*, *num_x_per_img*, *num_x_per_ref_img*)

Computing the similarity scores between *x* and *ref_x*.

参数

- **x** (*Tensor*) –of shape [N, C, H, W]. N is the number of key frame proposals.
- **ref_x** (*Tensor*) –of shape [M, C, H, W]. M is the number of reference frame proposals.
- **num_x_per_img** (*list[int]*) –The *x* contains proposals of multi-images. *num_x_per_img* denotes the number of proposals for each key image.
- **num_x_per_ref_img** (*list[int]*) –The *ref_x* contains proposals of multi-images. *num_x_per_ref_img* denotes the number of proposals for each reference image.

返回 The predicted similarity_logits of each pair of key image and reference image.

返回类型 list[Tensor]

get_targets(*sampling_results*, *gt_instance_ids*, *ref_gt_instance_ids*)

Calculate the ground truth for all samples in a batch according to the sampling_results.

参数

- **(List[obj** (*sampling_results*) –SamplingResults]): Assign results of all images in a batch after sampling.
- **gt_instance_ids** (*list[Tensor]*) –The instance ids of gt_bboxes of all images in a batch, each tensor has shape (num_gt,).
- **ref_gt_instance_ids** (*list[Tensor]*) –The instance ids of gt_bboxes of all reference images in a batch, each tensor has shape (num_gt,).

返回

Ground truth for proposals in a batch. Containing the following list of Tensors:

- track_id_targets (list[Tensor]): The instance ids of Gt_labels for all proposals in a batch, each tensor in list has shape (num_proposals,).
- track_id_weights (list[Tensor]): Labels_weights for all proposals in a batch, each tensor in list has shape (num_proposals,).

返回类型 Tuple[list[Tensor]]

loss(*similarity_logits*, *track_id_targets*, *track_id_weights*, *reduction_override=None*)

Calculate the loss in a batch.

参数

- **similarity_logits** (*list[Tensor]*) –The predicted similarity_logits of each pair of key image and reference image.
- **track_id_targets** (*list[Tensor]*) –The instance ids of Gt_labels for all proposals in a batch, each tensor in list has shape (num_proposals,).
- **track_id_weights** (*list[Tensor]*) –Labels_weights for all proposals in a batch, each tensor in list has shape (num_proposals,).
- **reduction_override** (*str, optional*) –The method used to reduce the loss. Options are "none", "mean" and "sum".

返回 a dictionary of loss components.

返回类型 dict[str, Tensor]

class mmtrack.models.track_heads.**RoITrackHead**(*roi_extractor=None*, *embed_head=None*, *regress_head=None*, *train_cfg=None*, *test_cfg=None*, *init_cfg=None*, **args*, ***kwargs*)

The roi track head.

This module is used in multi-object tracking methods, such as MaskTrack R-CNN.

参数

- **roi_extractor** (*dict*) –Configuration of roi extractor. Defaults to None.

- **embed_head** (*dict*) –Configuration of embed head. Defaults to None.
- **train_cfg** (*dict*) –Configuration when training. Defaults to None.
- **test_cfg** (*dict*) –Configuration when testing. Defaults to None.
- **init_cfg** (*dict*) –Configuration of initialization. Defaults to None.

extract_roi_feats(*x*, *bboxes*)

Extract roi features.

forward_train(*x*, *ref_x*, *img_metas*, *proposal_list*, *gt_bboxes*, *ref_gt_bboxes*, *gt_labels*, *gt_instance_ids*, *ref_gt_instance_ids*, *gt_bboxes_ignore=None*, ***kwargs*)

参数

- **x** (*list[Tensor]*) –list of multi-level image features.
- **ref_x** (*list[Tensor]*) –list of multi-level ref_img features.
- **img_metas** (*list[dict]*) –list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys see *mmtrack/datasets/pipelines/formatting.py:VideoCollect*.
- **proposal_list** (*list[Tensors]*) –list of region proposals.
- **gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for each image with shape (num_gts, 4) in [tl_x, tl_y, br_x, br_y] format.
- **ref_gt_bboxes** (*list[Tensor]*) –Ground truth bboxes for each reference image with shape (num_gts, 4) in [tl_x, tl_y, br_x, br_y] format.
- **gt_labels** (*list[Tensor]*) –class indices corresponding to each box.
- **gt_instance_ids** (*None | list[Tensor]*) –specify the instance id for each ground truth bbox.
- **ref_gt_instance_ids** (*None | list[Tensor]*) –specify the instance id for each ground truth bbox of reference images.
- **gt_bboxes_ignore** (*None | list[Tensor]*) –specify which bounding boxes can be ignored when computing the loss.

返回 a dictionary of loss components

返回类型 dict[str, Tensor]

init_assigner_sampler()

Initialize assigner and sampler.

init_embed_head(*roi_extractor*, *embed_head*)

Initialize `embed_head`

simple_test(*roi_feats*, *prev_roi_feats*)

Test without augmentations.

property with_track

whether the mulit-object tracker has a embed head

Type bool

class mmtrack.models.track_heads.**SiameseRPNHead**(*anchor_generator*, *in_channels*, *kernel_size=3*, *norm_cfg={'type': 'BN'}*, *weighted_sum=False*, *bbox_coder={'target_means': [0.0, 0.0, 0.0, 0.0], 'target_stds': [1.0, 1.0, 1.0, 1.0], 'type': 'DeltaXYWHBBoxCoder'}*, *loss_cls={'loss_weight': 1.0, 'reduction': 'sum', 'type': 'CrossEntropyLoss'}*, *loss_bbox={'loss_weight': 1.2, 'reduction': 'sum', 'type': 'L1Loss'}*, *train_cfg=None*, *test_cfg=None*, *init_cfg=None*, **args*, ***kwargs*)

Siamese RPN head.

This module is proposed in "SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks. SiamRPN++.

参数

- **anchor_generator** (*dict*) –Configuration to build anchor generator module.
- **in_channels** (*int*) –Input channels.
- **kernel_size** (*int*) –Kernel size of convs. Defaults to 3.
- **norm_cfg** (*dict*) –Configuration of normlization method after each conv. Defaults to dict(type='BN').
- **weighted_sum** (*bool*) –If True, use learnable weights to weightedly sum the output of multi heads in siamese rpn , otherwise, use averaging. Defaults to False.
- **bbox_coder** (*dict*) –Configuration to build bbox coder. Defaults to dict(type='DeltaXYWHBBoxCoder', target_means=[0., 0., 0., 0.], target_stds=[1., 1., 1., 1.]).
- **loss_cls** (*dict*) –Configuration to build classification loss. Defaults to dict(type='CrossEntropyLoss', reduction='sum', loss_weight=1.0)
- **loss_bbox** (*dict*) –Configuration to build bbox regression loss. Defaults to dict(type='L1Loss', reduction='sum', loss_weight=1.2).
- **train_cfg** (*Dict*) –Training setting. Defaults to None.
- **test_cfg** (*Dict*) –Testing setting. Defaults to None.

- **init_cfg** (*dict or list[dict], optional*) –Initialization config dict. Defaults to None.

forward(*z_feats*, *x_feats*)

Forward with features *z_feats* of exemplar images and features *x_feats* of search images.

参数

- **z_feats** (*tuple[Tensor]*) –Tuple of Tensor with shape (N, C, H, W) denoting the multi level feature maps of exemplar images. Typically H and W equal to 7.
- **x_feats** (*tuple[Tensor]*) –Tuple of Tensor with shape (N, C, H, W) denoting the multi level feature maps of search images. Typically H and W equal to 31.

返回 cls_score is a Tensor with shape (N, 2 * num_base_anchors, H, W), bbox_pred is a Tensor with shape (N, 4 * num_base_anchors, H, W), Typically H and W equal to 25.

返回类型 tuple(cls_score, bbox_pred)

get_bbox(*cls_score*, *bbox_pred*, *prev_bbox*, *scale_factor*)

Track *prev_bbox* to current frame based on the output of network.

参数

- **cls_score** (*Tensor*) –of shape (1, 2 * num_base_anchors, H, W).
- **bbox_pred** (*Tensor*) –of shape (1, 4 * num_base_anchors, H, W).
- **prev_bbox** (*Tensor*) –of shape (4,) in [cx, cy, w, h] format.
- **scale_factor** (*Tensr*) –scale factor.

返回 best_score is a Tensor denoting the score of *best_bbox*, best_bbox is a Tensor of shape (4,) with [cx, cy, w, h] format, which denotes the best tracked bbox in current frame.

返回类型 tuple(best_score, best_bbox)

get_targets(*gt_bboxes*, *score_maps_size*, *is_positive_pairs*)

Generate the training targets for exemplar image and search image pairs.

参数

- **gt_bboxes** (*list[Tensor]*) –Ground truth bboxes of each search image with shape (1, 5) in [0.0, tl_x, tl_y, br_x, br_y] format.
- **score_maps_size** (*torch.size*) –denoting the output size (height, width) of the network.
- **is_positive_pairs** (*bool*) –list of bool denoting whether each ground truth bbox in *gt_bboxes* is positive.

返回 tuple(all_labels, all_labels_weights, all_bbox_targets, all_bbox_weights): the shape is (N, H * W * num_base_anchors), (N, H * W * num_base_anchors), (N, H * W *

num_base_anchors, 4), (N, H * W * num_base_anchors, 4), respectively. All of them are Tensor.

loss(*cls_score*, *bbox_pred*, *labels*, *labels_weights*, *bbox_targets*, *bbox_weights*)

Compute loss.

参数

- **cls_score** (*Tensor*) –of shape (N, 2 * num_base_anchors, H, W).
- **bbox_pred** (*Tensor*) –of shape (N, 4 * num_base_anchors, H, W).
- **labels** (*Tensor*) –of shape (N, H * W * num_base_anchors).
- **labels_weights** (*Tensor*) –of shape (N, H * W * num_base_anchors).
- **bbox_targets** (*Tensor*) –of shape (N, H * W * num_base_anchors, 4).
- **bbox_weights** (*Tensor*) –of shape (N, H * W * num_base_anchors, 4).

返回 a dictionary of loss components

返回类型 dict[str, Tensor]

class mmtrack.models.track_heads.**StarkHead**(*num_query=1*, *transformer=None*, *positional_encoding={'normalize': True, 'num_feats': 128, 'type': 'SinePositionalEncoding'}*, *bbox_head=None*, *cls_head=None*, *loss_cls={'loss_weight': 1.0, 'type': 'CrossEntropyLoss', 'use_sigmoid': False}*, *loss_bbox={'loss_weight': 5.0, 'type': 'L1Loss'}*, *loss_iou={'loss_weight': 2.0, 'type': 'GIoULoss'}*, *train_cfg=None*, *test_cfg=None*, *init_cfg=None*, *frozen_modules=None*, ***kwargs*)

STARK head module for bounding box regression and prediction of confidence score of tracking bbox.

This module is proposed in "Learning Spatio-Temporal Transformer for Visual Tracking". STARK.

参数

- **num_query** (*int*) –Number of query in transformer.
- **(obj** (*test_cfg*) –'mmcv.ConfigDict'|dict): Config for transformer. Default: None.
- **(obj** –'mmcv.ConfigDict'|dict): Config for position encoding.
- **(obj** –'mmcv.ConfigDict'|dict, optional): Config for bbox head. Defaults to None.
- **(obj** –'mmcv.ConfigDict'|dict, optional): Config for classification head. Defaults to None.
- **(obj** –*mmcv.ConfigDict'|dict): Config of the classification loss. Default 'CrossEntropyLoss.*
- **(obj** –*mmcv.ConfigDict'|dict): Config of the bbox regression loss. Default 'L1Loss.*

- **(obj** –*mmcv.ConfigDict'\dict): Config of the bbox regression iou loss. Default 'GIoULoss.*
- **(obj** –'mmcv.ConfigDict'\ldict): Training config of transformer head.
- **(obj** –'mmcv.ConfigDict'\ldict): Testing config of transformer head.
- **init_cfg** (*dict or list[dict], optional*) –Initialization config dict. Default: None

forward(*inputs*)

" :param inputs: The list contains the

multi-level features and masks of template or search images.

- **'feat' : (tuple(Tensor)), the Tensor is of shape** (bs, c, h//stride, w//stride).
- 'mask' : (Tensor), of shape (bs, h, w).

Here, *h* and *w* denote the height and width of input image respectively. *stride* is the stride of feature map.

返回

- **'pred_bboxes' : (Tensor) of shape (bs, num_query, 4), in** [tl_x, tl_y, br_x, br_y] format
- ' pred_logit' : (Tensor) of shape (bs, num_query, 1)

返回类型 (dict)

forward_bbox_head(*feat*, *enc_mem*)

参数

- **feat** –output embeddings of decoder, with shape (1, bs, num_query, c).
- **enc_mem** –output embeddings of encoder, with shape (feats_flatten_len, bs, C)

  Here, 'feats_flatten_len' = z_feat_h*z_feat_w*2 + x_feat_h*x_feat_w. 'z_feat_h' and 'z_feat_w' denote the height and width of the template features respectively. 'x_feat_h' and 'x_feat_w' denote the height and width of search features respectively.

返回

of shape (bs, num_query, 4). The bbox is in [tl_x, tl_y, br_x, br_y] format.

返回类型 Tensor

init_weights()

Parameters initialization.

loss(*track_results*, *gt_bboxes*, *gt_labels*, *img_size=None*)

Compute loss.

参数

- **track_results** (*dict*) –it may contains the following keys: - ‘pred_bboxes’ : bboxes of (N, num_query, 4) shape in

  [tl_x, tl_y, br_x, br_y] format.

  - ’ pred_logits’ : bboxes of (N, num_query, 1) shape.

- **gt_bboxes** (*list[Tensor]*) –ground truth bboxes for search images with shape (N, 5) in [0., tl_x, tl_y, br_x, br_y] format.
- **gt_labels** (*list[Tensor]*) –ground truth labels for search images with shape (N, 2).
- **img_size** (*tuple, optional*) –the size (h, w) of original search image. Defaults to None.

返回 a dictionary of loss components.

返回类型 dict[str, Tensor]

26.11 builder

mmtrack.models.**build_aggregator**(*cfg*)

Build aggregator model.

mmtrack.models.**build_model**(*cfg*, *train_cfg=None*, *test_cfg=None*)

Build model.

mmtrack.models.**build_motion**(*cfg*)

Build motion model.

mmtrack.models.**build_reid**(*cfg*)

Build reid model.

mmtrack.models.**build_tracker**(*cfg*)

Build tracker.

CHAPTER 27

mmtrack.utils

mmtrack.utils.**collect_env**()

Collect the information of the running environments.

mmtrack.utils.**get_root_logger**(*log_file=None*, *log_level=20*)

Get root logger.

参数

- **log_file** (*str*) –File path of log. Defaults to None.
- **log_level** (*int*) –The level of logger. Defaults to logging.INFO.

返回 The obtained logger

返回类型 logging.Logger

CHAPTER 28

Indices and tables

- genindex
- search

Python 模块索引

m

索引

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

模块