mmcv Documentation

发布 *2.0.0*

MMCV Contributors

2023 年 08 月 30 日

介绍与安装

您可以在页面左下角切换中英文文档。

CHAPTER 1

介绍 MMCV

MMCV 是一个面向计算机视觉的基础库，它提供了以下功能：

- 图像和视频处理
- 图像和标注结果可视化
- 图像变换
- 多种 *CNN* 网络结构
- 高质量实现的常见 *CUDA* 算子

MMCV 支持多种平台，包括：

- Linux
- Windows
- macOS

它支持的 OpenMMLab 项目：

- MMClassification: OpenMMLab 图像分类工具箱
- MMDetection: OpenMMLab 目标检测工具箱
- MMDetection3D: OpenMMLab 新一代通用 3D 目标检测平台
- MMRotate: OpenMMLab 旋转框检测工具箱与测试基准
- MMYOLO: OpenMMLab YOLO 系列工具箱与测试基准
- MMSegmentation: OpenMMLab 语义分割工具箱

- MMOCR: OpenMMLab 全流程文字检测识别理解工具箱
- MMPose: OpenMMLab 姿态估计工具箱
- MMHuman3D: OpenMMLab 人体参数化模型工具箱与测试基准
- MMSelfSup: OpenMMLab 自监督学习工具箱与测试基准
- MMRazor: OpenMMLab 模型压缩工具箱与测试基准
- MMFewShot: OpenMMLab 少样本学习工具箱与测试基准
- MMAction2: OpenMMLab 新一代视频理解工具箱
- MMTracking: OpenMMLab 一体化视频目标感知平台
- MMFlow: OpenMMLab 光流估计工具箱与测试基准
- MMEditing: OpenMMLab 图像视频编辑工具箱
- MMGeneration: OpenMMLab 图片视频生成模型工具箱
- MMDeploy: OpenMMLab 模型部署框架

CHAPTER 2

安装 MMCV

MMCV 有两个版本：

- **mmcv**: 完整版，包含所有的特性以及丰富的开箱即用的 CPU 和 CUDA 算子。注意，完整版本可能需要更长时间来编译。
- **mmcv-lite**: 精简版，不包含 CPU 和 CUDA 算子但包含其余所有特性和功能，类似 MMCV 1.0 之前的版本。如果你不需要使用算子的话，精简版可以作为一个考虑选项。

警告： 请不要在同一个环境中安装两个版本，否则可能会遇到类似 `ModuleNotFound` 的错误。在安装一个版本之前，需要先卸载另一个。如果 `CUDA 可用，强烈推荐安装 mmcv`。

2.1 安装 mmcv

在安装 mmcv 之前，请确保 PyTorch 已经成功安装在环境中，可以参考 PyTorch 官方安装文档。可使用以下命令验证

```
python -c 'import torch;print(torch.__version__)'
```

如果输出版本信息，则表示 PyTorch 已安装。

2.1.1 使用 mim 安装（推荐）

mim 是 OpenMMLab 项目的包管理工具，使用它可以很方便地安装 mmcv。

```
pip install -U openmim
mim install "mmcv>=2.0.0rc1"
```

如果发现上述的安装命令没有使用预编译包（以 `.whl` 结尾）而是使用源码包（以 `.tar.gz` 结尾）安装，则有可能是我们没有提供和当前环境的 PyTorch 版本、CUDA 版本相匹配的 mmcv 预编译包，此时，你可以*源码安装 mmcv*。

Looking in links: https://download.openmmlab.com/mmcv/dist/cu102/torch1.8.0/index.html Collecting mmcv Downloading https://download.openmmlab.com/mmcv/dist/cu102/torch1.8.0/mmcv-2.0.0rc3-cp38-cp38-manylinux1_x86_64.whl

Looking in links: https://download.openmmlab.com/mmcv/dist/cu102/torch1.8.0/index.html Collecting mmcv==2.0.0rc3 Downloading mmcv-2.0.0rc3.tar.gz

如需安装指定版本的 mmcv，例如安装 2.0.0rc3 版本的 mmcv，可使用以下命令

```
mim install mmcv==2.0.0rc3
```

注解： 如果你打算使用 `opencv-python-headless` 而不是 `opencv-python`，例如在一个很小的容器环境或者没有图形用户界面的服务器中，你可以先安装 `opencv-python-headless`，这样在安装 mmcv 依赖的过程中会跳过 `opencv-python`。

另外，如果安装依赖库的时间过长，可以指定 pypi 源

```
mim install "mmcv>=2.0.0rc1" -i https://pypi.tuna.tsinghua.edu.cn/simple
```

安装完成后可以运行 check_installation.py 脚本检查 mmcv 是否安装成功。

2.1.2 使用 pip 安装

使用以下命令查看 CUDA 和 PyTorch 的版本

```
python -c 'import torch;print(torch.__version__);print(torch.version.cuda)'
```

根据系统的类型、CUDA 版本、PyTorch 版本以及 MMCV 版本选择相应的安装命令

如果在上面的下拉框中没有找到对应的版本，则可能是没有对应 PyTorch 或者 CUDA 或者 mmcv 版本的预编译包，此时，你可以*源码安装 mmcv*。

注解： PyTorch 在 1.x.0 和 1.x.1 之间通常是兼容的，故 mmcv 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。例如，如果你的 PyTorch 版本是 1.8.1，你可以放心选择 1.8.x。

注解： 如果你打算使用 `opencv-python-headless` 而不是 `opencv-python`，例如在一个很小的容器环境或者没有图形用户界面的服务器中，你可以先安装 `opencv-python-headless`，这样在安装 mmcv 依赖的过程中会跳过 `opencv-python`。

另外，如果安装依赖库的时间过长，可以指定 pypi 源

```
pip install "mmcv>=2.0.0rc1" -f https://download.openmmlab.com/mmcv/dist/cu111/torch1.
↪9.0/index.html -i https://pypi.tuna.tsinghua.edu.cn/simple
```

安装完成后可以运行 check_installation.py 脚本检查 mmcv 是否安装成功。

2.1.3 使用 docker 镜像

先将算法库克隆到本地再构建镜像

```
git clone https://github.com/open-mmlab/mmcv.git && cd mmcv
docker build -t mmcv -f docker/release/Dockerfile .
```

也可以直接使用下面的命令构建镜像

```
docker build -t mmcv https://github.com/open-mmlab/mmcv.git#2.x:docker/release
```

Dockerfile 默认安装最新的 mmcv，如果你想要指定版本，可以使用下面的命令

```
docker image build -t mmcv -f docker/release/Dockerfile --build-arg MMCV=2.0.0rc1 .
```

如果你想要使用其他版本的 PyTorch 和 CUDA，你可以在构建镜像时指定它们的版本。

例如指定 PyTorch 的版本是 1.11，CUDA 的版本是 11.3

```
docker build -t mmcv -f docker/release/Dockerfile \
    --build-arg PYTORCH=1.11.0 \
    --build-arg CUDA=11.3 \
    --build-arg CUDNN=8 \
    --build-arg MMCV=2.0.0rc1 .
```

更多 PyTorch 和 CUDA 镜像可以点击 dockerhub/pytorch 查看。

2.2 安装 mmcv-lite

如果你需要使用和 PyTorch 相关的模块，请确保 PyTorch 已经成功安装在环境中，可以参考 PyTorch 官方安装文档。

```
pip install mmcv-lite
```

CHAPTER 3

从源码编译 MMCV

3.1 编译 mmcv

在编译 mmcv 之前，请确保 PyTorch 已经成功安装在环境中，可以参考 PyTorch 官方安装文档。可使用以下命令验证

```
python -c 'import torch;print(torch.__version__)'
```

注解:

- 如果克隆代码仓库的速度过慢，可以使用以下命令克隆（注意：gitee 的 mmcv 不一定和 github 的保持一致，因为每天只同步一次）

```
git clone https://gitee.com/open-mmlab/mmcv.git
```

- 如果打算使用 `opencv-python-headless` 而不是 `opencv-python`，例如在一个很小的容器环境或者没有图形用户界面的服务器中，你可以先安装 `opencv-python-headless`，这样在安装 mmcv 依赖的过程中会跳过 `opencv-python`。
- 如果编译过程安装依赖库的时间过长，可以设置 pypi 源

```
pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
```

3.1.1 在 Linux 上编译 mmcv

| TODO: 视频教程

1. 克隆代码仓库

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

2. 安装 `ninja` 和 `psutil` 以加快编译速度

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

3. 检查 nvcc 的版本（要求大于等于 9.2，如果没有 GPU，可以跳过）

```
nvcc --version
```

上述命令如果输出以下信息，表示 nvcc 的设置没有问题，否则需要设置 CUDA_HOME

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Mon_Nov_30_19:08:53_PST_2020
Cuda compilation tools, release 11.2, V11.2.67
Build cuda_11.2.r11.2/compiler.29373293_0
```

注解： 如果想要支持 ROCm，可以参考 AMD ROCm 安装 ROCm。

---

4. 检查 gcc 的版本（要求大于等于 **5.4**）

```
gcc --version
```

5. 开始编译（预估耗时 10 分钟）

```
pip install -e . -v
```

6. 验证安装

```
python .dev_scripts/check_installation.py
```

如果上述命令没有报错，说明安装成功。如有报错，请查看问题解决页面是否已经有解决方案。

如果没有找到解决方案，欢迎提 issue。

3.1.2 在 macOS 上编译 mmcv

| TODO: 视频教程

注解： 如果你使用的是搭载 apple silicon 的 mac 设备，请安装 PyTorch 1.13+ 的版本，否则会遇到 issues#2218 中的问题。

---

1. 克隆代码仓库

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

2. 安装 `ninja` 和 `psutil` 以加快编译速度

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

3. 开始编译

```
pip install -e .
```

4. 验证安装

```
python .dev_scripts/check_installation.py
```

如果上述命令没有报错，说明安装成功。如有报错，请查看问题解决页面是否已经有解决方案。

如果没有找到解决方案，欢迎提 issue。

3.1.3 在 Windows 上编译 mmcv

| TODO: 视频教程

在 Windows 上编译 mmcv 比 Linux 复杂，本节将一步步介绍如何在 Windows 上编译 mmcv。

依赖项

请先安装以下的依赖项：

- Git：安装期间，请选择 **add git to Path**
- Visual Studio Community 2019：用于编译 C++ 和 CUDA 代码
- Miniconda：包管理工具
- CUDA 10.2：如果只需要 CPU 版本可以不安装 CUDA，安装 CUDA 时，可根据需要进行自定义安装。如果已经安装新版本的显卡驱动，建议取消驱动程序的安装

注解： 如果不清楚如何安装以上依赖，请参考Windows 环境从零安装 mmcv。另外，你需要知道如何在 Windows 上设置变量环境，尤其是“PATH”的设置，以下安装过程都会用到。

通用步骤

1. 从 Windows 菜单启动 Anaconda 命令行

   如 Miniconda 安装程序建议，不要使用原始的 `cmd.exe` 或是 `powershell.exe`。命令行有两个版本，一个基于 PowerShell，一个基于传统的 `cmd.exe`。请注意以下说明都是使用的基于 PowerShell

2. 创建一个新的 Conda 环境

```
(base) PS C:\Users\xxx> conda create --name mmcv python=3.7
(base) PS C:\Users\xxx> conda activate mmcv  # 确保做任何操作前先激活环境
```

3. 安装 PyTorch 时，可以根据需要安装支持 CUDA 或不支持 CUDA 的版本

```
# CUDA version
(mmcv) PS C:\Users\xxx> conda install pytorch torchvision cudatoolkit=10.2 -c␣
↪pytorch
# CPU version
(mmcv) PS C:\Users\xxx> conda install install pytorch torchvision cpuonly -c␣
↪pytorch
```

4. 克隆代码仓库

```
(mmcv) PS C:\Users\xxx> git clone https://github.com/open-mmlab/mmcv.git
(mmcv) PS C:\Users\xxx> cd mmcv
```

5. 安装 `ninja` 和 `psutil` 以加快编译速度

```
(mmcv) PS C:\Users\xxx\mmcv> pip install -r requirements/optional.txt
```

6. 设置 MSVC 编译器

   设置环境变量。添加 `C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.27.29110\bin\Hostx86\x64` 到 PATH，则 `cl.exe` 可以在命令行中运行，如下所示。

```
(mmcv) PS C:\Users\xxx\mmcv> cl
Microsoft (R) C/C++ Optimizing  Compiler Version 19.27.29111 for x64
Copyright (C) Microsoft Corporation.   All rights reserved.

usage: cl [ option... ] filename... [ / link linkoption... ]
```

为了兼容性，我们使用 x86-hosted 以及 x64-targeted 版本，即路径中的 `Hostx86\x64` 。

因为 PyTorch 将解析 `cl.exe` 的输出以检查其版本，只有 utf-8 将会被识别，你可能需要将系统语言更改为英语。控制面板 -> 地区-> 管理-> 非 Unicode 来进行语言转换。

编译与安装 mmcv

mmcv 有两个版本：

- 只包含 CPU 算子的版本

  编译 CPU 算子，但只有 x86 将会被编译，并且编译版本只能在 CPU only 情况下运行

- 既包含 CPU 算子，又包含 CUDA 算子的版本

  同时编译 CPU 和 CUDA 算子，`ops` 模块的 x86 与 CUDA 的代码都可以被编译。同时编译的版本可以在 CUDA 上调用 GPU

CPU 版本

编译安装

```
(mmcv) PS C:\Users\xxx\mmcv> python setup.py build_ext  # 如果成功, cl 将被启动用于编译算子
(mmcv) PS C:\Users\xxx\mmcv> python setup.py develop  # 安装
```

GPU 版本

1. 检查 `CUDA_PATH` 或者 `CUDA_HOME` 环境变量已经存在在 `envs` 之中

   ```
   (mmcv) PS C:\Users\xxx\mmcv> ls env:

   Name                           Value
   ----                           -----
   CUDA_PATH                      C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\
   ↪v10.2
   CUDA_PATH_V10_1                C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\
   ↪v10.1
   CUDA_PATH_V10_2                C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\
   ↪v10.2
   ```

   如果没有，你可以按照下面的步骤设置

```
(mmcv) PS C:\Users\xxx\mmcv> $env:CUDA_HOME = "C:\Program Files\NVIDIA GPU␣
↪Computing Toolkit\CUDA\v10.2"
# 或者
(mmcv) PS C:\Users\xxx\mmcv> $env:CUDA_HOME = $env:CUDA_PATH_V10_2  # CUDA_PATH_
↪V10_2 已经在环境变量中
```

2. 设置 CUDA 的目标架构

```
# 这里需要改成你的显卡对应的目标架构
(mmcv) PS C:\Users\xxx\mmcv> $env:TORCH_CUDA_ARCH_LIST="7.5"
```

注解： 可以点击 cuda-gpus 查看 GPU 的计算能力，也可以通过 CUDA 目录下的 deviceQuery.exe 工具查看

```
(mmcv) PS C:\Users\xxx\mmcv> &"C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\
↪v10.2\extras\demo_suite\deviceQuery.exe"
Device 0: "NVIDIA GeForce GTX 1660 SUPER"
CUDA Driver Version / Runtime Version          11.7 / 11.1
CUDA Capability Major/Minor version number:    7.5
```

上面的 7.5 表示目标架构。注意：需把上面命令的 v10.2 换成你的 CUDA 版本。

---

3. 编译安装

```
(mmcv) PS C:\Users\xxx\mmcv> python setup.py build_ext  # 如果成功，cl 将被启动用于编
译算子
(mmcv) PS C:\Users\xxx\mmcv> python setup.py develop # 安装
```

注解： 如果你的 PyTorch 版本是 1.6.0，你可能会遇到一些 issue 提到的错误，你可以参考这个 pull request 修改本地环境的 PyTorch 源代码

验证安装

```
(mmcv) PS C:\Users\xxx\mmcv> python .dev_scripts/check_installation.py
```

如果上述命令没有报错，说明安装成功。如有报错，请查看问题解决页面是否已经有解决方案。如果没有找到解决方案，欢迎提 issue。

3.2 编译 mmcv-lite

如果你需要使用和 PyTorch 相关的模块，请确保 PyTorch 已经成功安装在环境中，可以参考 PyTorch 官方安装文档。

1. 克隆代码仓库

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

2. 开始编译

```
MMCV_WITH_OPS=0 pip install -e . -v
```

3. 验证安装

```
python -c 'import mmcv;print(mmcv.__version__)'
```

CHAPTER 4

解读文章汇总

这篇文章汇总了 OpenMMLab 解读的部分文章（更多文章和视频见 OpenMMLabCourse），如果您有推荐的文章（不一定是 OpenMMLab 发布的文章，可以是自己写的文章），非常欢迎提 Pull Request 添加到这里。

4.1 MMCV 解读文章

4.1.1 框架解读

- MMCV 核心组件分析 (一)：整体概述
- MMCV 核心组件分析 (二)：FileHandler
- MMCV 核心组件分析 (三): FileClient
- MMCV 核心组件分析 (四): Config
- MMCV 核心组件分析 (五): Registry
- MMCV 核心组件分析 (六): Hook
- MMCV 核心组件分析 (七): Runner
- MMCV Hook 食用指南
- PyTorch & MMCV Dispatcher 机制解析

4.1.2 工具解读

- 训练可视化工具哪款是你的菜？MMCV 一行代码随你挑

4.1.3 安装指南

- 久等了！Windows 平台 MMCV 的预编译包终于来了！
- Windows 环境从零安装 mmcv-full

4.1.4 知乎问答

- 深度学习科研，如何高效进行代码和实验管理?
- 深度学习方面的科研工作中的实验代码有什么规范和写作技巧？如何妥善管理实验数据?

4.2 下游算法库解读文章

- MMDetection

4.3 PyTorch 解读文章

- PyTorch1.11 亮点一览：TorchData、functorch、DDP 静态图
- PyTorch1.12 亮点一览：DataPipe + TorchArrow 新的数据加载与处理范式
- PyTorch 源码解读之 nn.Module：核心网络模块接口详解
- PyTorch 源码解读之 torch.autograd：梯度计算详解
- PyTorch 源码解读之 torch.utils.data：解析数据处理全流程
- PyTorch 源码解读之 torch.optim：优化算法接口详解
- PyTorch 源码解读之 DP & DDP：模型并行和分布式训练解析
- PyTorch 源码解读之 BN & SyncBN：BN 与多卡同步 BN 详解
- PyTorch 源码解读之 torch.cuda.amp: 自动混合精度详解
- PyTorch 源码解读之 cpp_extension：揭秘 C++/CUDA 算子实现和调用全流程
- PyTorch 源码解读之即时编译篇
- PyTorch 源码解读之分布式训练了解一下?
- PyTorch 源码解读之 torch.serialization & torch.hub

4.4 其他

- 困扰我 48 小时的深拷贝，今天终于…
- 拿什么拯救我的 4G 显卡
- 是谁偷偷动了我的 logger
- 三句话，让 logger 言听计从
- Logging 不为人知的二三事
- Type Hints 入门教程，让代码更加规范整洁
- 手把手教你如何高效地在 MMCV 中贡献算子
- OpenMMLab 支持 IPU 训练芯片
- 基于 MMCV 走上开源大佬之路?

CHAPTER 5

数据处理

5.1 图像

图像模块提供了一些图像预处理的函数，该模块依赖 opencv 。

5.1.1 读取/保存/显示

使用 imread 和 imwrite 函数可以读取和保存图像。

```
import mmcv

img = mmcv.imread('test.jpg')
img = mmcv.imread('test.jpg', flag='grayscale')
img_ = mmcv.imread(img)  # 相当于什么也没做
mmcv.imwrite(img, 'out.jpg')
```

从二进制中读取图像

```
with open('test.jpg', 'rb') as f:
    data = f.read()
img = mmcv.imfrombytes(data)
```

显示图像文件或已读取的图像

```
mmcv.imshow('tests/data/color.jpg')

for i in range(10):
    img = np.random.randint(256, size=(100, 100, 3), dtype=np.uint8)
    mmcv.imshow(img, win_name='test image', wait_time=200)
```

5.1.2 色彩空间转换

支持的转换函数：

- bgr2gray
- gray2bgr
- bgr2rgb
- rgb2bgr
- bgr2hsv
- hsv2bgr

```
img = mmcv.imread('tests/data/color.jpg')
img1 = mmcv.bgr2rgb(img)
img2 = mmcv.rgb2gray(img1)
img3 = mmcv.bgr2hsv(img)
```

5.1.3 缩放

有三种缩放图像的方法。所有以 `imresize_*` 开头的函数都有一个 `return_scale` 参数，如果该参数为 `False`，函数的返回值只有调整之后的图像，否则是一个元组 `(resized_img, scale)`。

```
# 缩放图像至给定的尺寸
mmcv.imresize(img, (1000, 600), return_scale=True)

# 缩放图像至与给定的图像同样的尺寸
mmcv.imresize_like(img, dst_img, return_scale=False)

# 以一定的比例缩放图像
mmcv.imrescale(img, 0.5)

# 缩放图像至最长的边不大于 1000、最短的边不大于 800 并且没有改变图像的长宽比
mmcv.imrescale(img, (1000, 800))
```

5.1.4 旋转

我们可以使用 imrotate 旋转图像一定的角度。旋转的中心需要指定，默认值是原始图像的中心。有两种旋转的模式，一种保持图像的尺寸不变，因此旋转后原始图像中的某些部分会被裁剪，另一种是扩大图像的尺寸进而保留完整的原始图像。

```
img = mmcv.imread('tests/data/color.jpg')

# 顺时针旋转图像 30 度
img_ = mmcv.imrotate(img, 30)

# 逆时针旋转图像 90 度
img_ = mmcv.imrotate(img, -90)

# 顺时针旋转图像 30 度并且缩放图像为原始图像的 1.5 倍
img_ = mmcv.imrotate(img, 30, scale=1.5)

# 以坐标 (100, 100) 为中心顺时针旋转图像 30 度
img_ = mmcv.imrotate(img, 30, center=(100, 100))

# 顺时针旋转图像 30 度并扩大图像的尺寸
img_ = mmcv.imrotate(img, 30, auto_bound=True)
```

5.1.5 翻转

我们可以使用 imflip 翻转图像。

```
img = mmcv.imread('tests/data/color.jpg')

# 水平翻转图像
mmcv.imflip(img)

# 垂直翻转图像
mmcv.imflip(img, direction='vertical')
```

5.1.6 裁剪

imcrop 可以裁剪图像的一个或多个区域，每个区域用左上角和右下角坐标表示，形如 (x1, y1, x2, y2)

```
import mmcv
import numpy as np
```

(下页继续)

(续上页)

```
img = mmcv.imread('tests/data/color.jpg')

# 裁剪区域 (10, 10, 100, 120)
bboxes = np.array([10, 10, 100, 120])
patch = mmcv.imcrop(img, bboxes)

# 裁剪两个区域，分别是 (10, 10, 100, 120) 和 (0, 0, 50, 50)
bboxes = np.array([[10, 10, 100, 120], [0, 0, 50, 50]])
patches = mmcv.imcrop(img, bboxes)

# 裁剪两个区域并且缩放区域 1.2 倍
patches = mmcv.imcrop(img, bboxes, scale=1.2)
```

5.1.7 填充

impad and impad_to_multiple 可以用给定的值将图像填充至给定的尺寸。

```
img = mmcv.imread('tests/data/color.jpg')

# 用给定值将图像填充至 (1000, 1200)
img_ = mmcv.impad(img, shape=(1000, 1200), pad_val=0)

# 用给定值分别填充图像的 3 个通道至 (1000, 1200)
img_ = mmcv.impad(img, shape=(1000, 1200), pad_val=(100, 50, 200))

# 用给定值填充图像的左、右、上、下四条边
img_ = mmcv.impad(img, padding=(10, 20, 30, 40), pad_val=0)

# 用 3 个值分别填充图像的左、右、上、下四条边的 3 个通道
img_ = mmcv.impad(img, padding=(10, 20, 30, 40), pad_val=(100, 50, 200))

# 将图像的四条边填充至能够被给定值整除
img_ = mmcv.impad_to_multiple(img, 32)
```

5.2 视频

视频模块提供了以下的功能：

- 一个 VideoReader 类，具有友好的 API 接口可以读取和转换视频
- 一些编辑视频的方法，包括 cut ， concat ， resize
- 光流的读取/保存/变换

5.2.1 VideoReader

VideoReader 类提供了和序列一样的接口去获取视频帧。该类会缓存所有被访问过的帧。

```
video = mmcv.VideoReader('test.mp4')

# 获取基本的信息
print(len(video))
print(video.width, video.height, video.resolution, video.fps)

# 遍历所有的帧
for frame in video:
    print(frame.shape)

# 读取下一帧
img = video.read()

# 使用索引获取帧
img = video[100]

# 获取指定范围的帧
img = video[5:10]
```

将视频切成帧并保存至给定目录或者从给定目录中生成视频。

```
# 将视频切成帧并保存至目录
video = mmcv.VideoReader('test.mp4')
video.cvt2frames('out_dir')

# 从给定目录中生成视频
mmcv.frames2video('out_dir', 'test.avi')
```

5.2.2 编辑函数

有几个用于编辑视频的函数，这些函数是对 ffmpeg 的封装。

```
# 裁剪视频
mmcv.cut_video('test.mp4', 'clip1.mp4', start=3, end=10, vcodec='h264')

# 将多个视频拼接成一个视频
mmcv.concat_video(['clip1.mp4', 'clip2.mp4'], 'joined.mp4', log_level='quiet')

# 将视频缩放至给定的尺寸
mmcv.resize_video('test.mp4', 'resized1.mp4', (360, 240))
```

(下页继续)

(续上页)

```
# 将视频缩放至给定的倍率
mmcv.resize_video('test.mp4', 'resized2.mp4', ratio=2)
```

5.2.3 光流

mmcv 提供了以下用于操作光流的函数：

- 读取/保存
- 可视化
- 流变换

我们提供了两种将光流 dump 到文件的方法，分别是非压缩和压缩的方法。非压缩的方法直接将浮点数值的光流保存至二进制文件，虽然光流无损但文件会比较大。而压缩的方法先量化光流至 0-255 整形数值再保存为 jpeg 图像。光流的 x 维度和 y 维度会被拼接到图像中。

1. 读取/保存

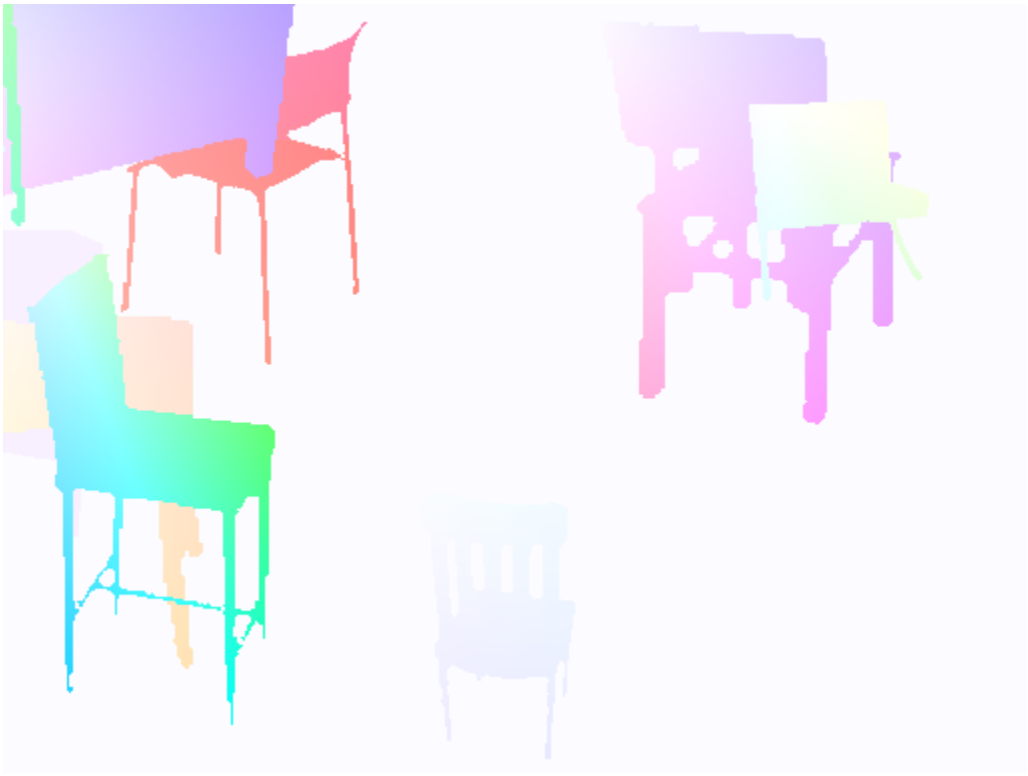
```
flow = np.random.rand(800, 600, 2).astype(np.float32)
# 保存光流到 flo 文件 (~3.7M)
mmcv.flowwrite(flow, 'uncompressed.flo')
# 保存光流为 jpeg 图像 (~230K)，图像的尺寸为 (800, 1200)
mmcv.flowwrite(flow, 'compressed.jpg', quantize=True, concat_axis=1)

# 读取光流文件，以下两种方式读取的光流尺寸均为 (800, 600, 2)
flow = mmcv.flowread('uncompressed.flo')
flow = mmcv.flowread('compressed.jpg', quantize=True, concat_axis=1)
```

2. 可视化

使用 mmcv.flowshow() 可视化光流

```
mmcv.flowshow(flow)
```



1. 流变换

```
img1 = mmcv.imread('img1.jpg')
flow = mmcv.flowread('flow.flo')
warped_img2 = mmcv.flow_warp(img1, flow)
```

img1 (左) and img2 (右)

光流 (img2 -> img1)

变换后的图像和真实图像的差异

CHAPTER 6

数据变换

在 OpenMMLab 算法库中，数据集的构建和数据的准备是相互解耦的。通常，数据集的构建只对数据集进行解析，记录每个样本的基本信息；而数据的准备则是通过一系列的数据变换，根据样本的基本信息进行数据加载、预处理、格式化等操作。

6.1 数据变换的设计

在 MMCV 中，我们使用各种可调用的数据变换类来进行数据的操作。这些数据变换类可以接受若干配置参数进行实例化，之后通过调用的方式对输入的数据字典进行处理。同时，我们约定所有数据变换都接受一个字典作为输入，并将处理后的数据输出为一个字典。一个简单的例子如下：

```
>>> import numpy as np
>>> from mmcv.transforms import Resize
>>>
>>> transform = Resize(scale=(224, 224))
>>> data_dict = {'img': np.random.rand(256, 256, 3)}
>>> data_dict = transform(data_dict)
>>> print(data_dict['img'].shape)
(224, 224, 3)
```

数据变换类会读取输入字典的某些字段，并且可能添加、或者更新某些字段。这些字段的键大部分情况下是固定的，如 `Resize` 会固定地读取输入字典中的 `"img"` 等字段。我们可以在对应类的文档中了解对输入输出字段的约定。

注解： 默认情况下，在需要图像尺寸作为**初始化参数**的数据变换 (如 Resize, Pad) 中，图像尺寸的顺序均为 (width, height)。在数据变换**返回的字典**中，图像相关的尺寸，如 `img_shape`、`ori_shape`、`pad_shape` 等，均为 (height, width)。

MMCV 为所有的数据变换类提供了一个统一的基类 (`BaseTransform`)：

```
class BaseTransform(metaclass=ABCMeta):

    def __call__(self, results: dict) -> dict:

        return self.transform(results)

    @abstractmethod
    def transform(self, results: dict) -> dict:
        pass
```

所有的数据变换类都需要继承 `BaseTransform`，并实现 `transform` 方法。`transform` 方法的输入和输出均为一个字典。在**自定义数据变换类**一节中，我们会更详细地介绍如何实现一个数据变换类。

6.2 数据流水线

如上所述，所有数据变换的输入和输出都是一个字典，而且根据 OpenMMLab 中 有关数据集的约定，数据集中每个样本的基本信息都是一个字典。这样一来，我们可以将所有的数据变换操作首尾相接，组合成为一条数据流水线（data pipeline），输入数据集中样本的信息字典，输出完成一系列处理后的信息字典。

以分类任务为例，我们在下图展示了一个典型的数据流水线。对每个样本，数据集中保存的基本信息是一个如图中最左侧所示的字典，之后每经过一个由蓝色块代表的数据变换操作，数据字典中都会加入新的字段（标记为绿色）或更新现有的字段（标记为橙色）。

在配置文件中，数据流水线是一个若干数据变换配置字典组成的列表，每个数据集都需要设置参数 `pipeline` 来定义该数据集需要进行的数据准备操作。如上数据流水线在配置文件中的配置如下：

```
pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='Resize', size=256, keep_ratio=True),
    dict(type='CenterCrop', crop_size=224),
    dict(type='Normalize', mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.
↪375]),
    dict(type='ClsFormatBundle')
]

dataset = dict(
```

(下页继续)

(续上页)

```
    ...
    pipeline=pipeline,
    ...
)
```

6.3 常用的数据变换类

按照功能，常用的数据变换类可以大致分为数据加载、数据预处理与增强、数据格式化。在 MMCV 中，我们提供了一些常用的数据变换类如下：

6.3.1 数据加载

为了支持大规模数据集的加载，通常在 `Dataset` 初始化时不加载数据，只加载相应的路径。因此需要在数据流水线中进行具体数据的加载。

6.3.2 数据预处理及增强

数据预处理和增强通常是对图像本身进行变换，如裁剪、填充、缩放等。

6.3.3 数据格式化

数据格式化操作通常是对数据进行的类型转换。

6.4 自定义数据变换类

要实现一个新的数据变换类，需要继承 `BaseTransform`，并实现 `transform` 方法。这里，我们使用一个简单的翻转变换（`MyFlip`）作为示例：

```python
import random
import mmcv
from mmcv.transforms import BaseTransform, TRANSFORMS

@TRANSFORMS.register_module()
class MyFlip(BaseTransform):
    def __init__(self, direction: str):
        super().__init__()
        self.direction = direction
```

(下页继续)

(续上页)

```
    def transform(self, results: dict) -> dict:
        img = results['img']
        results['img'] = mmcv.imflip(img, direction=self.direction)
        return results
```

从而，我们可以实例化一个 MyFlip 对象，并将之作为一个可调用对象，来处理我们的数据字典。

```
import numpy as np

transform = MyFlip(direction='horizontal')
data_dict = {'img': np.random.rand(224, 224, 3)}
data_dict = transform(data_dict)
processed_img = data_dict['img']
```

又或者，在配置文件的 pipeline 中使用 MyFlip 变换

```
pipeline = [
    ...
    dict(type='MyFlip', direction='horizontal'),
    ...
]
```

需要注意的是，如需在配置文件中使用，需要保证 MyFlip 类所在的文件在运行时能够被导入。

6.5 变换包装

变换包装是一种特殊的数据变换类，他们本身并不操作数据字典中的图像、标签等信息，而是对其中定义的数据变换的行为进行增强。

6.5.1 字段映射（KeyMapper）

字段映射包装（KeyMapper）用于对数据字典中的字段进行映射。例如，一般的图像处理变换都从数据字典中的 "img" 字段获得值。但有些时候，我们希望这些变换处理数据字典中其他字段中的图像，比如 "gt_img" 字段。

如果配合注册器和配置文件使用的话，在配置文件中数据集的 pipeline 中如下例使用字段映射包装：

```
pipeline = [
    ...
    dict(type='KeyMapper',
        mapping={
            'img': 'gt_img',  # 将 "gt_img" 字段映射至 "img" 字段
```

(下页继续)

(续上页)

```
            'mask': ...,  # 不使用原始数据中的 "mask" 字段。即对于被包装的数据变换，数据中不包含
↪"mask" 字段
        },
        auto_remap=True,  # 在完成变换后，将 "img" 重映射回 "gt_img" 字段
        transforms=[
            # 在 `RandomFlip` 变换类中，我们只需要操作 "img" 字段即可
            dict(type='RandomFlip'),
        ])
    ...
]
```

利用字段映射包装，我们在实现数据变换类时，不需要考虑在 `transform` 方法中考虑各种可能的输入字段名，只需要处理默认的字段即可。

6.5.2 随机选择（RandomChoice）和随机执行（RandomApply）

随机选择包装（`RandomChoice`）用于从一系列数据变换组合中随机应用一个数据变换组合。利用这一包装，我们可以简单地实现一些数据增强功能，比如 AutoAugment。

如果配合注册器和配置文件使用的话，在配置文件中数据集的 `pipeline` 中如下例使用随机选择包装：

```
pipeline = [
    ...
    dict(type='RandomChoice',
        transforms=[
            [
                dict(type='Posterize', bits=4),
                dict(type='Rotate', angle=30.)
            ],  # 第一种随机变化组合
            [
                dict(type='Equalize'),
                dict(type='Rotate', angle=30)
            ],  # 第二种随机变换组合
        ],
        prob=[0.4, 0.6]  # 两种随机变换组合各自的选用概率
        )
    ...
]
```

随机执行包装（`RandomApply`）用于以指定概率随机执行数据变换组合。例如：

```
pipeline = [
    ...
```

(下页继续)

(续上页)

```
    dict(type='RandomApply',
        transforms=[dict(type='Rotate', angle=30.)],
        prob=0.3)  # 以 0.3 的概率执行被包装的数据变换
    ...
]
```

6.5.3 多目标扩展（TransformBroadcaster）

通常，一个数据变换类只会从一个固定的字段读取操作目标。虽然我们也可以使用 KeyMapper 来改变读取的字段，但无法将变换一次性应用于多个字段的数据。为了实现这一功能，我们需要借助多目标扩展包装（TransformBroadcaster）。

多目标扩展包装（TransformBroadcaster）有两个用法，一是将数据变换作用于指定的多个字段，二是将数据变换作用于某个字段下的一组目标中。

1. 应用于多个字段

   假设我们需要将数据变换应用于 "lq" (low-quality) 和 "gt" (ground-truth) 两个字段中的图像上。

```
pipeline = [
    dict(type='TransformBroadcaster',
        # 分别应用于 "lq" 和 "gt" 两个字段，并将二者应设置 "img" 字段
        mapping={'img': ['lq', 'gt']},
        # 在完成变换后，将 "img" 字段重映射回原先的字段
        auto_remap=True,
        # 是否在对各目标的变换中共享随机变量
        # 更多介绍参加后续章节（随机变量共享）
        share_random_params=True,
        transforms=[
            # 在 `RandomFlip` 变换类中，我们只需要操作 "img" 字段即可
            dict(type='RandomFlip'),
        ])
]
```

   在多目标扩展的 mapping 设置中，我们同样可以使用 ... 来忽略指定的原始字段。如以下例子中，被包裹的 RandomCrop 会对字段 "img" 中的图像进行裁剪，并且在字段 "img_shape" 存在时更新剪裁后的图像大小。如果我们希望同时对两个图像字段 "lq" 和 "gt" 进行相同的随机裁剪，但只更新一次 "img_shape" 字段，可以通过例子中的方式实现：

```
pipeline = [
    dict(type='TransformBroadcaster',
        mapping={
            'img': ['lq', 'gt'],
            'img_shape': ['img_shape', ...],
```

(下页继续)

(续上页)

```
        },
        # 在完成变换后，将 "img" 和 "img_shape" 字段重映射回原先的字段
        auto_remap=True,
        # 是否在对各目标的变换中共享随机变量
        # 更多介绍参加后续章节（随机变量共享）
        share_random_params=True,
        transforms=[
            # `RandomCrop` 类中会操作 "img" 和 "img_shape" 字段。若 "img_shape" 空缺，
            # 则只操作 "img"
            dict(type='RandomCrop'),
        ])
]
```

2. 应用于一个字段的一组目标

   假设我们需要将数据变换应用于 "images" 字段，该字段为一个图像组成的 list。

```
pipeline = [
    dict(type='TransformBroadcaster',
        # 将 "images" 字段下的每张图片映射至 "img" 字段
        mapping={'img': 'images'},
        # 在完成变换后，将 "img" 字段下的图片重映射回 "images" 字段的列表中
        auto_remap=True,
        # 是否在对各目标的变换中共享随机变量
        share_random_params=True,
        transforms=[
            # 在 `RandomFlip` 变换类中，我们只需要操作 "img" 字段即可
            dict(type='RandomFlip'),
        ])
]
```

装饰器 cache_randomness

在 TransformBroadcaster 中，我们提供了 share_random_params 选项来支持在多次数据变换中共享随机状态。例如，在超分辨率任务中，我们希望将随机变换**同步**作用于低分辨率图像和原始图像。如果我们希望在自定义的数据变换类中使用这一功能，需要在类中标注哪些随机变量是支持共享的。这可以通过装饰器 cache_randomness 来实现。

以上文中的 MyFlip 为例，我们希望以一定的概率随机执行翻转：

```
from mmcv.transforms.utils import cache_randomness

@TRANSFORMS.register_module()
```

(下页继续)

(续上页)

```
class MyRandomFlip(BaseTransform):
    def __init__(self, prob: float, direction: str):
        super().__init__()
        self.prob = prob
        self.direction = direction

    @cache_randomness  # 标注该方法的输出为可共享的随机变量
    def do_flip(self):
        flip = True if random.random() > self.prob else False
        return flip

    def transform(self, results: dict) -> dict:
        img = results['img']
        if self.do_flip():
            results['img'] = mmcv.imflip(img, direction=self.direction)
        return results
```

在上面的例子中，我们用 cache_randomness 装饰 do_flip 方法，即将该方法返回值 flip 标注为一个支持共享的随机变量。进而，在 TransformBroadcaster 对多个目标的变换中，这一变量的值都会保持一致。

装饰器 avoid_cache_randomness

在一些情况下，我们无法将数据变换中产生随机变量的过程单独放在类方法中。例如数据变换中使用的来自第三方库的模块，这些模块将随机变量相关的部分封装在了内部，导致无法将其抽出为数据变换的类方法。这样的数据变换无法通过装饰器 cache_randomness 标注支持共享的随机变量，进而无法在多目标扩展时共享随机变量。

为了避免在多目标扩展中误用此类数据变换，我们提供了另一个装饰器 avoid_cache_randomness，用来对此类数据变换进行标记：

```
from mmcv.transforms.utils import avoid_cache_randomness

@TRANSFORMS.register_module()
@avoid_cache_randomness
class MyRandomTransform(BaseTransform):

    def transform(self, results: dict) -> dict:
        ...
```

用 avoid_cache_randomness 标记的数据变换类，当其实例被 TransformBroadcaster 包装且将参数 share_random_params 设置为 True 时，会抛出异常，以此提醒用户不能这样使用。

在使用 avoid_cache_randomness 时需要注意以下几点：

1. `avoid_cache_randomness` 只用于装饰数据变换类（BaseTransfrom 的子类），而不能用与装饰其他一般的类、类方法或函数
2. 被 `avoid_cache_randomness` 修饰的数据变换作为基类时，其子类将**不会继承**这一特性。如果子类仍无法共享随机变量，则应再次使用 `avoid_cache_randomness` 修饰
3. 只有当一个数据变换具有随机性，且无法共享随机参数时，才需要以 `avoid_cache_randomness` 修饰。无随机性的数据变换不需要修饰

CHAPTER 7

可视化

mmcv 可以展示图像以及标注（目前只支持标注框）

```
# 展示图像文件
mmcv.imshow('a.jpg')

# 展示已加载的图像
img = np.random.rand(100, 100, 3)
mmcv.imshow(img)

# 展示带有标注框的图像
img = np.random.rand(100, 100, 3)
bboxes = np.array([[0, 0, 50, 50], [20, 20, 60, 60]])
mmcv.imshow_bboxes(img, bboxes)
```

mmcv 也可以展示特殊的图像，例如光流

```
flow = mmcv.flowread('test.flo')
mmcv.flowshow(flow)
```

CHAPTER 8

卷积神经网络

我们为卷积神经网络提供了一些构建模块，包括层构建、模块组件和权重初始化。

8.1 网络层的构建

在运行实验时，我们可能需要尝试同属一种类型但不同配置的层，但又不希望每次都修改代码。于是我们提供一些层构建方法，可以从字典构建层，字典可以在配置文件中配置，也可以通过命令行参数指定。

8.1.1 用法

一个简单的例子：

```
from mmcv.cnn import build_conv_layer

cfg = dict(type='Conv3d')
layer = build_conv_layer(cfg, in_channels=3, out_channels=8, kernel_size=3)
```

- `build_conv_layer`: 支持的类型包括 Conv1d、Conv2d、Conv3d、Conv (Conv 是 Conv2d 的别名）
- `build_norm_layer`: 支持的类型包括 BN1d、BN2d、BN3d、BN (alias for BN2d)、SyncBN、GN、LN、IN1d、IN2d、IN3d、IN（IN 是 IN2d 的别名）
- `build_activation_layer`：支持的类型包括 ReLU、LeakyReLU、PReLU、RReLU、ReLU6、ELU、Sigmoid、Tanh、GELU

- `build_upsample_layer`: 支持的类型包括 nearest、bilinear、deconv、pixel_shuffle
- `build_padding_layer`: 支持的类型包括 zero、reflect、replicate

8.1.2 拓展

我们还允许自定义层和算子来扩展构建方法。

1. 编写和注册自己的模块：

```
from mmengine.registry import MODELS

@MODELS.register_module()
class MyUpsample:

    def __init__(self, scale_factor):
        pass

    def forward(self, x):
        pass
```

2. 在某处导入 `MyUpsample`（例如 `__init__.py`）然后使用它：

```
from mmcv.cnn import build_upsample_layer

cfg = dict(type='MyUpsample', scale_factor=2)
layer = build_upsample_layer(cfg)
```

8.2 模块组件

我们还提供了常用的模块组件，以方便网络构建。卷积组件 `ConvModule` 由 convolution、normalization 以及 activation layers 组成，更多细节请参考 ConvModule api。

```
from mmcv.cnn import ConvModule

# conv + bn + relu
conv = ConvModule(3, 8, 2, norm_cfg=dict(type='BN'))
# conv + gn + relu
conv = ConvModule(3, 8, 2, norm_cfg=dict(type='GN', num_groups=2))
# conv + relu
conv = ConvModule(3, 8, 2)
# conv
conv = ConvModule(3, 8, 2, act_cfg=None)
```

(下页继续)

(续上页)

```
# conv + leaky relu
conv = ConvModule(3, 8, 3, padding=1, act_cfg=dict(type='LeakyReLU'))
# bn + conv + relu
conv = ConvModule(
    3, 8, 2, norm_cfg=dict(type='BN'), order=('norm', 'conv', 'act'))
```

8.3 Model Zoo

除了 `torchvision` 的预训练模型，我们还提供以下 CNN 的预训练模型：

- VGG Caffe
- ResNet Caffe
- ResNeXt
- ResNet with Group Normalization
- ResNet with Group Normalization and Weight Standardization
- HRNetV2
- Res2Net
- RegNet

8.3.1 Model URLs in JSON

MMCV 中的 Model Zoo Link 由 JSON 文件管理。json 文件由模型名称及其 url 或 path 的键值对组成，一个 json 文件可能类似于：

```
{
    "model_a": "https://example.com/models/model_a_9e5bac.pth",
    "model_b": "pretrain/model_b_ab3ef2c.pth"
}
```

可以在此处找到托管在 OpenMMLab AWS 上的预训练模型的默认链接。

你可以通过将 `open-mmlab.json` 放在 `MMCV_HOME` 下来覆盖默认链接，如果在环境中找不到 `MMCV_HOME`，则默认使用 `~/.cache/mmcv`。当然你也可以使用命令 `export MMCV_HOME=/your/path` 来设置自己的路径。

外部的 json 文件将被合并为默认文件，如果相同的键出现在外部 `json` 和默认 `json` 中，则将使用外部 `json`。

8.3.2 Load Checkpoint

`mmcv.load_checkpoint()` 的参数 `filename` 支持以下类型：

- filepath: `checkpoint` 路径
- `http://xxx` and `https://xxx`: 下载 checkpoint 的链接，文件名中必需包含 `SHA256` 后缀
- `torchvision://xxx`: `torchvision.models` 中的模型链接，更多细节参考 torchvision
- `open-mmlab://xxx`: 默认和其他 json 文件中提供的模型链接或文件路径

CHAPTER 9

算子

MMCV 提供了检测、分割等任务中常用的算子

CHAPTER 10

English

CHAPTER 11

简体中文

CHAPTER 12

v1.3.18

部分自定义算子对于不同的设备有不同实现，为此添加的大量宏命令与类型检查使得代码变得难以维护。例如：

```
  if (input.device().is_cuda()) {
#ifdef MMCV_WITH_CUDA
    CHECK_CUDA_INPUT(input);
    CHECK_CUDA_INPUT(rois);
    CHECK_CUDA_INPUT(output);
    CHECK_CUDA_INPUT(argmax_y);
    CHECK_CUDA_INPUT(argmax_x);

    roi_align_forward_cuda(input, rois, output, argmax_y, argmax_x,
                           aligned_height, aligned_width, spatial_scale,
                           sampling_ratio, pool_mode, aligned);
#else
    AT_ERROR("RoIAlign is not compiled with GPU support");
#endif
  } else {
    CHECK_CPU_INPUT(input);
    CHECK_CPU_INPUT(rois);
    CHECK_CPU_INPUT(output);
    CHECK_CPU_INPUT(argmax_y);
    CHECK_CPU_INPUT(argmax_x);
    roi_align_forward_cpu(input, rois, output, argmax_y, argmax_x,
```

(下页继续)

(续上页)

```
                          aligned_height, aligned_width, spatial_scale,
                          sampling_ratio, pool_mode, aligned);
  }
```

为此我们设计了注册与分发的机制以更好的管理这些算子实现。

```
void ROIAlignForwardCUDAKernelLauncher(Tensor input, Tensor rois, Tensor output,
                                       Tensor argmax_y, Tensor argmax_x,
                                       int aligned_height, int aligned_width,
                                       float spatial_scale, int sampling_ratio,
                                       int pool_mode, bool aligned);

void roi_align_forward_cuda(Tensor input, Tensor rois, Tensor output,
                            Tensor argmax_y, Tensor argmax_x,
                            int aligned_height, int aligned_width,
                            float spatial_scale, int sampling_ratio,
                            int pool_mode, bool aligned) {
  ROIAlignForwardCUDAKernelLauncher(
      input, rois, output, argmax_y, argmax_x, aligned_height, aligned_width,
      spatial_scale, sampling_ratio, pool_mode, aligned);
}

// 注册算子的 cuda 实现
void roi_align_forward_impl(Tensor input, Tensor rois, Tensor output,
                            Tensor argmax_y, Tensor argmax_x,
                            int aligned_height, int aligned_width,
                            float spatial_scale, int sampling_ratio,
                            int pool_mode, bool aligned);
REGISTER_DEVICE_IMPL(roi_align_forward_impl, CUDA, roi_align_forward_cuda);

// roi_align.cpp
// 使用 dispatcher 根据参数中的 Tensor device 类型对实现进行分发
void roi_align_forward_impl(Tensor input, Tensor rois, Tensor output,
                            Tensor argmax_y, Tensor argmax_x,
                            int aligned_height, int aligned_width,
                            float spatial_scale, int sampling_ratio,
                            int pool_mode, bool aligned) {
  DISPATCH_DEVICE_IMPL(roi_align_forward_impl, input, rois, output, argmax_y,
                       argmax_x, aligned_height, aligned_width, spatial_scale,
                       sampling_ratio, pool_mode, aligned);
}
```

CHAPTER 13

v1.3.11

为了灵活地支持更多的后端和硬件，例如 NVIDIA GPUs 、AMD GPUs，我们重构了 mmcv/ops/csrc 目录。注意，这次重构不会影响 API 的使用。更多相关信息，请参考 PR1206。

原始的目录结构如下所示

```
.
├── common_cuda_helper.hpp
├── ops_cuda_kernel.cuh
├── pytorch_cpp_helper.hpp
├── pytorch_cuda_helper.hpp
├── parrots_cpp_helper.hpp
├── parrots_cuda_helper.hpp
├── parrots_cudawarpfunction.cuh
├── onnxruntime
│   ├── onnxruntime_register.h
│   ├── onnxruntime_session_options_config_keys.h
│   ├── ort_mmcv_utils.h
│   ├── ...
│   ├── onnx_ops.h
│   └── cpu
│       ├── onnxruntime_register.cpp
│       ├── ...
│       └── onnx_ops_impl.cpp
├── parrots
│   ├── ...
```

(下页继续)

(续上页)

```
|   ├── ops.cpp
|   ├── ops_cuda.cu
|   ├── ops_parrots.cpp
|   └── ops_pytorch.h
├── pytorch
|   ├── ...
|   ├── ops.cpp
|   ├── ops_cuda.cu
|   ├── pybind.cpp
└── tensorrt
    ├── trt_cuda_helper.cuh
    ├── trt_plugin_helper.hpp
    ├── trt_plugin.hpp
    ├── trt_serialize.hpp
    ├── ...
    ├── trt_ops.hpp
    └── plugins
        ├── trt_cuda_helper.cu
        ├── trt_plugin.cpp
        ├── ...
        ├── trt_ops.cpp
        └── trt_ops_kernel.cu
```

重构之后，它的结构如下所示

```
.
├── common
|   ├── box_iou_rotated_utils.hpp
|   ├── parrots_cpp_helper.hpp
|   ├── parrots_cuda_helper.hpp
|   ├── pytorch_cpp_helper.hpp
|   ├── pytorch_cuda_helper.hpp
|   └── cuda
|       ├── common_cuda_helper.hpp
|       ├── parrots_cudawarpfunction.cuh
|       ├── ...
|       └── ops_cuda_kernel.cuh
├── onnxruntime
|   ├── onnxruntime_register.h
|   ├── onnxruntime_session_options_config_keys.h
|   ├── ort_mmcv_utils.h
|   ├── ...
|   ├── onnx_ops.h
|   └── cpu
```

(下页继续)

(续上页)

```
|       ├── onnxruntime_register.cpp
|       ├── ...
|       └── onnx_ops_impl.cpp
├── parrots
|   ├── ...
|   ├── ops.cpp
|   ├── ops_parrots.cpp
|   └── ops_pytorch.h
├── pytorch
|   ├── info.cpp
|   ├── pybind.cpp
|   ├── ...
|   ├── ops.cpp
|   └── cuda
|       ├── ...
|       └── ops_cuda.cu
└── tensorrt
    ├── trt_cuda_helper.cuh
    ├── trt_plugin_helper.hpp
    ├── trt_plugin.hpp
    ├── trt_serialize.hpp
    ├── ...
    ├── trt_ops.hpp
    └── plugins
        ├── trt_cuda_helper.cu
        ├── trt_plugin.cpp
        ├── ...
        ├── trt_ops.cpp
        └── trt_ops_kernel.cu
```

CHAPTER 14

常见问题

在这里我们列出了用户经常遇到的问题以及对应的解决方法。如果您遇到了其他常见的问题，并且知道可以帮到大家的解决办法，欢迎随时丰富这个列表。

14.1 安装问题

- KeyError: “xxx: ‘yyy is not in the zzz registry’”

  只有模块所在的文件被导入时，注册机制才会被触发，所以您需要在某处导入该文件，更多详情请查看 KeyError: “MaskRCNN: ‘RefineRoIHead is not in the models registry’” 。

- “No module named ‘mmcv.ops’”；“No module named ‘mmcv._ext’”

  1. 使用 `pip uninstall mmcv` 卸载您环境中的 mmcv
  2. 参考 installation instruction 或者 Build MMCV from source 安装 mmcv-full

- “invalid device function”或者“no kernel image is available for execution”

  1. 检查 GPU 的 CUDA 计算能力
  2. 运行 `python mmdet/utils/collect_env.py` 来检查 PyTorch、torchvision 和 MMCV 是否是针对正确的 GPU 架构构建的，您可能需要去设置 `TORCH_CUDA_ARCH_LIST` 来重新安装 MMCV。兼容性问题可能会出现在使用旧版的 GPUs，如：colab 上的 Tesla K80 (3.7)
  3. 检查运行环境是否和 mmcv/mmdet 编译时的环境相同。例如，您可能使用 CUDA 10.0 编译 mmcv，但在 CUDA 9.0 的环境中运行它

- “undefined symbol”或者“cannot open xxx.so”

1. 如果符号和 CUDA/C++ 相关（例如：libcudart.so 或者 GLIBCXX），请检查 CUDA/GCC 运行时的版本是否和编译 mmcv 的一致
2. 如果符号和 PyTorch 相关（例如：符号包含 caffe、aten 和 TH），请检查 PyTorch 运行时的版本是否和编译 mmcv 的一致
3. 运行 `python mmdet/utils/collect_env.py` 以检查 PyTorch、torchvision 和 MMCV 构建和运行的环境是否相同

- "RuntimeError: CUDA error: invalid configuration argument"

  这个错误可能是由于您的 GPU 性能不佳造成的。尝试降低 THREADS_PER_BLOCK 的值并重新编译 mmcv。

- "RuntimeError: nms is not compiled with GPU support"

  这个错误是由于您的 CUDA 环境没有正确安装。您可以尝试重新安装您的 CUDA 环境，然后删除 mmcv/build 文件夹并重新编译 mmcv。

- "Segmentation fault"

  1. 检查 GCC 的版本，通常是因为 PyTorch 版本与 GCC 版本不匹配（例如 GCC < 4.9 ），我们推荐用户使用 GCC 5.4，我们也不推荐使用 GCC 5.5，因为有反馈 GCC 5.5 会导致 "segmentation fault" 并且切换到 GCC 5.4 就可以解决问题
  2. 检查是否正确安装 CUDA 版本的 PyTorc。输入以下命令并检查是否返回 True

     ```
     python -c 'import torch; print(torch.cuda.is_available())'
     ```

  3. 如果 `torch` 安装成功，那么检查 MMCV 是否安装成功。输入以下命令，如果没有报错说明 mmcv-full 安装成。

     ```
     python -c 'import mmcv; import mmcv.ops'
     ```

  4. 如果 MMCV 与 PyTorch 都安装成功了，则可以使用 `ipdb` 设置断点或者使用 `print` 函数，分析是哪一部分的代码导致了 `segmentation fault`

- "libtorch_cuda_cu.so: cannot open shared object file"

  `mmcv-full` 依赖 `libtorch_cuda_cu.so` 文件，但程序运行时没能找到该文件。我们可以检查该文件是否存在 `~/miniconda3/envs/{environment-name}/lib/python3.7/site-packages/torch/lib` 也可以尝试重装 PyTorch。

- "fatal error C1189: #error: –unsupported Microsoft Visual Studio version!"

  如果您在 Windows 上编译 mmcv-full 并且 CUDA 的版本是 9.2，您很可能会遇到这个问题 `"C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.2\include\crt/host_config.h(133): fatal error C1189: #error: -- unsupported Microsoft Visual Studio version! Only the versions 2012, 2013, 2015 and 2017 are supported!"`，您可以尝试使用低版本的 Microsoft Visual Studio，例如 vs2017。

- “error: member “torch::jit::detail::ModulePolicy::all_slots” may not be initialized”

  如果您在 Windows 上编译 mmcv-full 并且 PyTorch 的版本是 1.5.0，您很可能会遇到这个问题 `- torch/csrc/jit/api/module.h(474): error: member "torch::jit::detail::ModulePolicy::all_slots" may not be initialized`。 解决这个问题的方法是将 `torch/csrc/jit/api/module.h` 文件中所有 `static constexpr bool all_slots = false;` 替换为 `static bool all_slots = false;`。更多细节可以查看 member “torch::jit::detail::AttributePolicy::all_slots” may not be initialized。

- “error: a member with an in-class initializer must be const”

  如果您在 Windows 上编译 mmcv-full 并且 PyTorch 的版本是 1.6.0，您很可能会遇到这个问题 `"- torch/include\torch/csrc/jit/api/module.h(483): error: a member with an in-class initializer must be const"`. 解决这个问题的方法是将 `torch/include\torch/csrc/jit/api/module.h` 文件中的所有 `CONSTEXPR_EXCEPT_WIN_CUDA` 替换为 `const`。更多细节可以查看 Ninja: build stopped: subcommand failed。

- “error: member “torch::jit::ProfileOptionalOp::Kind” may not be initialized”

  如果您在 Windows 上编译 mmcv-full 并且 PyTorch 的版本是 1.7.0，您很可能会遇到这个问题 `torch/include\torch/csrc/jit/ir/ir.h(1347): error: member "torch::jit::ProfileOptionalOp::Kind" may not be initialized`. 解决这个问题的方法是修改 PyTorch 中的几个文件：

  - 删除 `torch/include\torch/csrc/jit/ir/ir.h` 文件中的 `static constexpr Symbol Kind = ::c10::prim::profile;` 和 `tatic constexpr Symbol Kind = ::c10::prim::profile_optional;`
  - 将 `torch\include\pybind11\cast.h` 文件中的 `explicit operator type&() { return *(this->value); }` 替换为 `explicit operator type&() { return *((type*)this->value); }`
  - 将 `torch/include\torch/csrc/jit/api/module.h` 文件中的所有 `CONSTEXPR_EXCEPT_WIN_CUDA` 替换为 `const`

  更多细节可以查看 Ensure default extra_compile_args。

- MMCV 和 MMDetection 的兼容性问题；”ConvWS is already registered in conv layer”

  请参考 installation instruction 为您的 MMDetection 版本安装正确版本的 MMCV。

14.2 使用问题

- “RuntimeError: Expected to have finished reduction in the prior iteration before starting a new one”

  1. 这个错误是因为有些参数没有参与 loss 的计算，可能是代码中存在多个分支，导致有些分支没有参与 loss 的计算。更多细节见 Expected to have finished reduction in the prior iteration before starting a new one。
  2. 你可以设置 DDP 中的 `find_unused_parameters` 为 `True`，或者手动查找哪些参数没有用到。

- “RuntimeError: Trying to backward through the graph a second time”

  不能同时设置 `GradientCumulativeOptimizerHook` 和 `OptimizerHook`，这会导致 `loss.backward()` 被调用两次，于是程序抛出 `RuntimeError`。我们只需设置其中的一个。更多细节见 Trying to backward through the graph a second time。

CHAPTER 15

贡献代码

欢迎加入 MMCV 社区，我们致力于打造最前沿的计算机视觉基础库，我们欢迎任何类型的贡献，包括但不限于

修复错误

修复代码实现错误的步骤如下：

1. 如果提交的代码改动较大，建议先提交 issue，并正确描述 issue 的现象、原因和复现方式，讨论后确认修复方案。
2. 修复错误并补充相应的单元测试，提交拉取请求。

新增功能或组件

1. 如果新功能或模块涉及较大的代码改动，建议先提交 issue，确认功能的必要性。
2. 实现新增功能并添单元测试，提交拉取请求。

文档补充

修复文档可以直接提交拉取请求

添加文档或将文档翻译成其他语言步骤如下

1. 提交 issue，确认添加文档的必要性。
2. 添加文档，提交拉取请求。

15.1 拉取请求工作流

如果你对拉取请求不了解，没关系，接下来的内容将会从零开始，一步一步地指引你如何创建一个拉取请求。如果你想深入了解拉取请求的开发模式，可以参考 github 官方文档

15.1.1 1. 复刻仓库

当你第一次提交拉取请求时，先复刻 OpenMMLab 原代码库，点击 GitHub 页面右上角的 **Fork** 按钮，复刻后的代码库将会出现在你的 GitHub 个人主页下。

将代码克隆到本地

```
git clone git@github.com:{username}/mmcv.git
```

添加原代码库为上游代码库

```
git remote add upstream git@github.com:open-mmlab/mmcv
```

检查 remote 是否添加成功，在终端输入 `git remote -v`

```
origin        git@github.com:{username}/mmcv.git (fetch)
origin        git@github.com:{username}/mmcv.git (push)
upstream        git@github.com:open-mmlab/mmcv (fetch)
upstream        git@github.com:open-mmlab/mmcv (push)
```

注解： 这里对 origin 和 upstream 进行一个简单的介绍，当我们使用 git clone 来克隆代码时，会默认创建一个 origin 的 remote，它指向我们克隆的代码库地址，而 upstream 则是我们自己添加的，用来指向原始代码库地址。当然如果你不喜欢他叫 upstream，也可以自己修改，比如叫 open-mmlab。我们通常向 origin 提交代码（即 fork 下来的远程仓库），然后向 upstream 提交一个 pull request。如果提交的代码和最新的代码发生冲突，再从 upstream 拉取最新的代码，和本地分支解决冲突，再提交到 origin。

15.1.2 2. 配置 pre-commit

在本地开发环境中，我们使用 pre-commit 来检查代码风格，以确保代码风格的统一。在提交代码，需要先安装 pre-commit（需要在 MMCV 目录下执行）：

```
pip install -U pre-commit
pre-commit install
```

检查 pre-commit 是否配置成功，并安装 `.pre-commit-config.yaml` 中的钩子：

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

注解： 如果你是中国用户，由于网络原因，可能会出现安装失败的情况，这时可以使用国内源

pre-commit install -c .pre-commit-config-zh-cn.yaml

pre-commit run –all-files -c .pre-commit-config-zh-cn.yaml

如果安装过程被中断，可以重复执行 `pre-commit run ...` 继续安装。

如果提交的代码不符合代码风格规范，pre-commit 会发出警告，并自动修复部分错误。

如果我们想临时绕开 pre-commit 的检查提交一次代码，可以在 `git commit` 时加上 `--no-verify`（需要保证最后推送至远程仓库的代码能够通过 pre-commit 检查）。

```
git commit -m "xxx" --no-verify
```

15.1.3 3. 创建开发分支

安装完 pre-commit 之后，我们需要基于 master 创建开发分支，建议的分支命名规则为 `username/pr_name`。

```
git checkout -b yhc/refactor_contributing_doc
```

在后续的开发中，如果本地仓库的 master 分支落后于 upstream 的 master 分支，我们需要先拉取 upstream 的代码进行同步，再执行上面的命令

```
git pull upstream master
```

15.1.4 4. 提交代码并在本地通过单元测试

- MMCV 引入了 mypy 来做静态类型检查，以增加代码的鲁棒性。因此我们在提交代码时，需要补充 Type Hints。具体规则可以参考教程。
- 提交的代码同样需要通过单元测试

  ```
  # 通过全量单元测试
  pytest tests

  # 我们需要保证提交的代码能够通过修改模块的单元测试，以 runner 为例
  pytest tests/test_runner/test_runner.py
  ```

  如果你由于缺少依赖无法运行修改模块的单元测试，可以参考指引-单元测试

- 如果修改/添加了文档，参考指引确认文档渲染正常。

15.1.5 5. 推送代码到远程

代码通过单元测试和 pre-commit 检查后，将代码推送到远程仓库，如果是第一次推送，可以在 `git push` 后加上 `-u` 参数以关联远程分支

```
git push -u origin {branch_name}
```

这样下次就可以直接使用 `git push` 命令推送代码了，而无需指定分支和远程仓库。

15.1.6 6. 提交拉取请求（PR）

(1) 在 GitHub 的 Pull request 界面创建拉取请求

(2) 根据指引修改 PR 描述，以便于其他开发者更好地理解你的修改

描述规范详见拉取请求规范

注意事项

(a) PR 描述应该包含修改理由、修改内容以及修改后带来的影响，并关联相关 Issue（具体方式见文档）

(b) 如果是第一次为 OpenMMLab 做贡献，需要签署 CLA

(c) 检查提交的 PR 是否通过 CI（集成测试）

MMCV 会在不同的平台（Linux、Window、Mac），基于不同版本的 Python、PyTorch、CUDA 对提交的代码进行单元测试，以保证代码的正确性，如果有任何一个没有通过，我们可点击上图中的 `Details` 来查看具体的测试信息，以便于我们修改代码。

(3) 如果 PR 通过了 CI，那么就可以等待其他开发者的 review，并根据 reviewer 的意见，修改代码，并重复 4-*5* 步骤，直到 reviewer 同意合入 PR。

所有 reviewer 同意合入 PR 后，我们会尽快将 PR 合并到主分支。

15.1.7 7. 解决冲突

随着时间的推移，我们的代码库会不断更新，这时候，如果你的 PR 与主分支存在冲突，你需要解决冲突，解决冲突的方式有两种：

```
git fetch --all --prune
git rebase upstream/master
```

或者

```
git fetch --all --prune
git merge upstream/master
```

如果你非常善于处理冲突，那么可以使用 rebase 的方式来解决冲突，因为这能够保证你的 commit log 的整洁。如果你不太熟悉 `rebase` 的使用，那么可以使用 `merge` 的方式来解决冲突。

15.2 指引

15.2.1 单元测试

如果你无法正常执行部分模块的单元测试，例如 video 模块，可能是你的当前环境没有安装以下依赖

```
# Linux
sudo apt-get update -y
sudo apt-get install -y libturbojpeg
sudo apt-get install -y ffmpeg

# Windows
conda install ffmpeg
```

在提交修复代码错误或新增特性的拉取请求时，我们应该尽可能的让单元测试覆盖所有提交的代码，计算单元测试覆盖率的方法如下

```
python -m coverage run -m pytest /path/to/test_file
python -m coverage html
# check file in htmlcov/index.html
```

15.2.2 文档渲染

在提交修复代码错误或新增特性的拉取请求时，可能会需要修改/新增模块的 docstring。我们需要确认渲染后的文档样式是正确的。本地生成渲染后的文档的方法如下

```
pip install -r requirements/docs.txt
cd docs/zh_cn/
# or docs/en
make html
# check file in ./docs/zh_cn/_build/html/index.html
```

15.3 代码风格

15.3.1 Python

PEP8 作为 OpenMMLab 算法库首选的代码规范，我们使用以下工具检查和格式化代码

- flake8: Python 官方发布的代码规范检查工具，是多个检查工具的封装
- isort: 自动调整模块导入顺序的工具
- yapf: Google 发布的代码规范检查工具
- codespell: 检查单词拼写是否有误
- mdformat: 检查 markdown 文件的工具
- docformatter: 格式化 docstring 的工具

yapf 和 isort 的配置可以在 setup.cfg 找到

通过配置 pre-commit hook ，我们可以在提交代码时自动检查和格式化 `flake8`、`yapf`、`isort`、`trailing whitespaces`、`markdown files`，修复 `end-of-files`、`double-quoted-strings`、`python-encoding-pragma`、`mixed-line-ending`，调整 `requirments.txt` 的包顺序。pre-commit 钩子的配置可以在 .pre-commit-config 找到。

pre-commit 具体的安装使用方式见拉取请求。

更具体的规范请参考*OpenMMLab 代码规范*。

15.3.2 C++ and CUDA

C++ 和 CUDA 的代码规范遵从 Google C++ Style Guide

15.4 拉取请求规范

1. 使用 pre-commit hook，尽量减少代码风格相关问题
2. 一个拉取请求对应一个短期分支
3. 粒度要细，一个拉取请求只做一件事情，避免超大的拉取请求
   - Bad：实现 Faster R-CNN
   - Acceptable：给 Faster R-CNN 添加一个 box head
   - Good：给 box head 增加一个参数来支持自定义的 conv 层数
4. 每次 Commit 时需要提供清晰且有意义 commit 信息
5. 提供清晰且有意义的拉取请求描述

- 标题写明白任务名称，一般格式:[Prefix] Short description of the pull request (Suffix)
- prefix: 新增功能 [Feature], 修 bug [Fix], 文档相关 [Docs], 开发中 [WIP] (暂时不会被 review)
- 描述里介绍拉取请求的主要修改内容，结果，以及对其他部分的影响, 参考拉取请求模板
- 关联相关的议题 (issue) 和其他拉取请求

6. 如果引入了其他三方库，或借鉴了三方库的代码，请确认他们的许可证和 mmcv 兼容，并在借鉴的代码上补充 `This code is inspired from http://`

CHAPTER 16

拉取请求

本文档的内容已迁移到贡献指南。

CHAPTER 17

代码规范

17.1 代码规范标准

17.1.1 PEP 8 ——Python 官方代码规范

Python 官方的代码风格指南，包含了以下几个方面的内容：

- 代码布局，介绍了 Python 中空行、断行以及导入相关的代码风格规范。比如一个常见的问题：当我的代码较长，无法在一行写下时，何处可以断行？
- 表达式，介绍了 Python 中表达式空格相关的一些风格规范。
- 尾随逗号相关的规范。当列表较长，无法一行写下而写成如下逐行列表时，推荐在末项后加逗号，从而便于追加选项、版本控制等。

```
# Correct:
FILES = ['setup.cfg', 'tox.ini']
# Correct:
FILES = [
    'setup.cfg',
    'tox.ini',
]
# Wrong:
FILES = ['setup.cfg', 'tox.ini',]
# Wrong:
FILES = [
```

(下页继续)

(续上页)

```
    'setup.cfg',
    'tox.ini'
]
```

- 命名相关规范、注释相关规范、类型注解相关规范，我们将在后续章节中做详细介绍。

  “A style guide is about consistency. Consistency with this style guide is important. Consistency within a project is more important. Consistency within one module or function is the most important.” PEP 8 –Style Guide for Python Code

---

注解： PEP 8 的代码规范并不是绝对的，项目内的一致性要优先于 PEP 8 的规范。OpenMMLab 各个项目都在 setup.cfg 设定了一些代码规范的设置，请遵照这些设置。一个例子是在 PEP 8 中有如下一个例子：

```
# Correct:
hypot2 = x*x + y*y
# Wrong:
hypot2 = x * x + y * y
```

这一规范是为了指示不同优先级，但 OpenMMLab 的设置中通常没有启用 yapf 的 `ARITHMETIC_PRECEDENCE_INDICATION` 选项，因而格式规范工具不会按照推荐样式格式化，以设置为准。

17.1.2 Google 开源项目风格指南

Google 使用的编程风格指南，包括了 Python 相关的章节。相较于 PEP 8，该指南提供了更为详尽的代码指南。该指南包括了语言规范和风格规范两个部分。

其中，语言规范对 Python 中很多语言特性进行了优缺点的分析，并给出了使用指导意见，如异常、Lambda 表达式、列表推导式、metaclass 等。

风格规范的内容与 PEP 8 较为接近，大部分约定建立在 PEP 8 的基础上，也有一些更为详细的约定，如函数长度、TODO 注释、文件与 socket 对象的访问等。

推荐将该指南作为参考进行开发，但不必严格遵照，一来该指南存在一些 Python 2 兼容需求，例如指南中要求所有无基类的类应当显式地继承 Object, 而在仅使用 Python 3 的环境中，这一要求是不必要的，依本项目中的惯例即可。二来 OpenMMLab 的项目作为框架级的开源软件，不必对一些高级技巧过于避讳，尤其是 MMCV。但尝试使用这些技巧前应当认真考虑是否真的有必要，并寻求其他开发人员的广泛评估。

另外需要注意的一处规范是关于包的导入，在该指南中，要求导入本地包时必须使用路径全称，且导入的每一个模块都应当单独成行，通常这是不必要的，而且也不符合目前项目的开发惯例，此处进行如下约定：

```
# Correct
from mmcv.cnn.bricks import (Conv2d, build_norm_layer, DropPath, MaxPool2d,
```

(下页继续)

(续上页)

```
                              Linear)
from ..utils import ext_loader

# Wrong
from mmcv.cnn.bricks import Conv2d, build_norm_layer, DropPath, MaxPool2d, \
                            Linear  # 使用括号进行连接，而不是反斜杠
from ...utils import is_str  # 最多向上回溯一层，过多的回溯容易导致结构混乱
```

OpenMMLab 项目使用 pre-commit 工具自动格式化代码，详情见贡献代码。

17.2 命名规范

17.2.1 命名规范的重要性

优秀的命名是良好代码可读的基础。基础的命名规范对各类变量的命名做了要求，使读者可以方便地根据代码名了解变量是一个类 / 局部变量 / 全局变量等。而优秀的命名则需要代码作者对于变量的功能有清晰的认识，以及良好的表达能力，从而使读者根据名称就能了解其含义，甚至帮助了解该段代码的功能。

17.2.2 基础命名规范

注意：

- 尽量避免变量名与保留字冲突，特殊情况下如不可避免，可使用一个后置下划线，如 class_
- 尽量不要使用过于简单的命名，除了约定俗成的循环变量 i，文件变量 f，错误变量 e 等。
- 不会被用到的变量可以命名为 _，逻辑检查器会将其忽略。

17.2.3 命名技巧

良好的变量命名需要保证三点：

1. 含义准确，没有歧义
2. 长短适中
3. 前后统一

```
# Wrong
class Masks(metaclass=ABCMeta):  # 命名无法表现基类；Instance or Semantic？
    pass

# Correct
```

(下页继续)

(续上页)

```
class BaseInstanceMasks(metaclass=ABCMeta):
    pass

# Wrong，不同地方含义相同的变量尽量用统一的命名
def __init__(self, inplanes, planes):
    pass

def __init__(self, in_channels, out_channels):
    pass
```

常见的函数命名方法：

- 动宾命名法：crop_img, init_weights
- 动宾倒置命名法：imread, bbox_flip

注意函数命名与参数的顺序，保证主语在前，符合语言习惯：

- check_keys_exist(key, container)
- check_keys_contain(container, key)

注意避免非常规或统一约定的缩写，如 nb -> num_blocks，in_nc -> in_channels

17.3 docstring 规范

17.3.1 为什么要写 docstring

docstring 是对一个类、一个函数功能与 API 接口的详细描述，有两个功能，一是帮助其他开发者了解代码功能，方便 debug 和复用代码；二是在 Readthedocs 文档中自动生成相关的 API reference 文档，帮助不了解源代码的社区用户使用相关功能。

17.3.2 如何写 docstring

与注释不同，一份规范的 docstring 有着严格的格式要求，以便于 Python 解释器以及 sphinx 进行文档解析，详细的 docstring 约定参见 PEP 257。此处以例子的形式介绍各种文档的标准格式，参考格式为 Google 风格。

1. 模块文档

   代码风格规范推荐为每一个模块（即 Python 文件）编写一个 docstring，但目前 OpenMMLab 项目大部分没有此类 docstring，因此不做硬性要求。

   ```
   """A one line summary of the module or program, terminated by a period.

   Leave one blank line. The rest of this docstring should contain an
   ```

(下页继续)

(续上页)

```
overall description of the module or program. Optionally, it may also
contain a brief description of exported classes and functions and/or usage
examples.

Typical usage example:

foo = ClassFoo()
bar = foo.FunctionBar()
"""
```

2. 类文档

   类文档是我们最常需要编写的，此处，按照 OpenMMLab 的惯例，我们使用了与 Google 风格不同的写法。如下例所示，文档中没有使用 Attributes 描述类属性，而是使用 Args 描述 **init** 函数的参数。

   在 Args 中，遵照 `parameter (type): Description.` 的格式，描述每一个参数类型和功能。其中，多种类型可使用 `(float or str)` 的写法，可以为 None 的参数可以写为 `(int, optional)`。

```
class BaseRunner(metaclass=ABCMeta):
    """The base class of Runner, a training helper for PyTorch.

    All subclasses should implement the following APIs:

    - ``run()``
    - ``train()``
    - ``val()``
    - ``save_checkpoint()``

    Args:
        model (:obj:`torch.nn.Module`): The model to be run.
        batch_processor (callable, optional): A callable method that process
            a data batch. The interface of this method should be
            ``batch_processor(model, data, train_mode) -> dict``.
            Defaults to None.
        optimizer (dict or :obj:`torch.optim.Optimizer`, optional): It can be
            either an optimizer (in most cases) or a dict of optimizers
            (in models that requires more than one optimizer, e.g., GAN).
            Defaults to None.
        work_dir (str, optional): The working directory to save checkpoints
            and logs. Defaults to None.
        logger (:obj:`logging.Logger`): Logger used during training.
             Defaults to None. (The default value is just for backward
             compatibility)
        meta (dict, optional): A dict records some import information such as
```

(下页继续)

(续上页)

```
            environment info and seed, which will be logged in logger hook.
            Defaults to None.
        max_epochs (int, optional): Total training epochs. Defaults to None.
        max_iters (int, optional): Total training iterations. Defaults to None.
    """

    def __init__(self,
                 model,
                 batch_processor=None,
                 optimizer=None,
                 work_dir=None,
                 logger=None,
                 meta=None,
                 max_iters=None,
                 max_epochs=None):
        ...
```

另外，在一些算法实现的主体类中，建议加入原论文的链接；如果参考了其他开源代码的实现，则应加入 modified from，而如果是直接复制了其他代码库的实现，则应加入 copied from ，并注意源码的 License。如有必要，也可以通过.. math:: 来加入数学公式

```
# 参考实现
# This func is modified from `detectron2
# <https://github.com/facebookresearch/detectron2/blob/
↪ffff8acc35ea88ad1cb1806ab0f00b4c1c5dbfd9/detectron2/structures/masks.py#L387>`_.

# 复制代码
# This code was copied from the `ubelt
# library<https://github.com/Erotemic/ubelt>`_.

# 引用论文 & 添加公式
class LabelSmoothLoss(nn.Module):
    r"""Initializer for the label smoothed cross entropy loss.

    Refers to `Rethinking the Inception Architecture for Computer Vision
    <https://arxiv.org/abs/1512.00567>`_.

    This decreases gap between output scores and encourages generalization.
    Labels provided to forward can be one-hot like vectors (NxC) or class
    indices (Nx1).
    And this accepts linear combination of one-hot like labels from mixup or
    cutmix except multi-label task.
```

(下页继续)

(续上页)

```
    Args:
        label_smooth_val (float): The degree of label smoothing.
        num_classes (int, optional): Number of classes. Defaults to None.
        mode (str): Refers to notes, Options are "original", "classy_vision",
            "multi_label". Defaults to "classy_vision".
        reduction (str): The method used to reduce the loss.
            Options are "none", "mean" and "sum". Defaults to 'mean'.
        loss_weight (float):  Weight of the loss. Defaults to 1.0.

    Note:
        if the ``mode`` is "original", this will use the same label smooth
        method as the original paper as:

        .. math::
            (1-\epsilon)\delta_{k, y} + \frac{\epsilon}{K}

        where :math:`\epsilon` is the ``label_smooth_val``, :math:`K` is
        the ``num_classes`` and :math:`\delta_{k,y}` is Dirac delta,
        which equals 1 for k=y and 0 otherwise.

        if the ``mode`` is "classy_vision", this will use the same label
        smooth method as the `facebookresearch/ClassyVision
        <https://github.com/facebookresearch/ClassyVision/blob/main/classy_vision/
↪losses/label_smoothing_loss.py>`_ repo as:

        .. math::
            \frac{\delta_{k, y} + \epsilon/K}{1+\epsilon}

        if the ``mode`` is "multi_label", this will accept labels from
        multi-label task and smoothing them as:

        .. math::
            (1-2\epsilon)\delta_{k, y} + \epsilon
```

注解： 注意 “here“、‘here‘、” here” 三种引号功能是不同。

在 reStructured 语法中，“here“ 表示一段代码；‘here‘ 表示斜体；” here” 无特殊含义，一般可用来表示字符串。其中 ‘here‘ 的用法与 Markdown 中不同，需要多加留意。另外还有:obj:‘type‘ 这种更规范的表示类的写法，但鉴于长度，不做特别要求，一般仅用于表示非常用类型。

3. 方法（函数）文档

函数文档与类文档的结构基本一致，但需要加入返回值文档。对于较为复杂的函数和类，可以使用

Examples 字段加入示例；如果需要对参数加入一些较长的备注，可以加入 Note 字段进行说明。

对于使用较为复杂的类或函数，比起看大段大段的说明文字和参数文档，添加合适的示例更能帮助用户迅速了解其用法。需要注意的是，这些示例最好是能够直接在 Python 交互式环境中运行的，并给出一些相对应的结果。如果存在多个示例，可以使用注释简单说明每段示例，也能起到分隔作用。

```
def import_modules_from_strings(imports, allow_failed_imports=False):
    """Import modules from the given list of strings.

    Args:
        imports (list | str | None): The given module names to be imported.
        allow_failed_imports (bool): If True, the failed imports will return
            None. Otherwise, an ImportError is raise. Defaults to False.

    Returns:
        List[module] | module | None: The imported modules.
        All these three lines in docstring will be compiled into the same
        line in readthedocs.

    Examples:
        >>> osp, sys = import_modules_from_strings(
        ...     ['os.path', 'sys'])
        >>> import os.path as osp_
        >>> import sys as sys_
        >>> assert osp == osp_
        >>> assert sys == sys_
    """
    ...
```

如果函数接口在某个版本发生了变化，需要在 docstring 中加入相关的说明，必要时添加 Note 或者 Warning 进行说明，例如：

```
class CheckpointHook(Hook):
    """Save checkpoints periodically.

    Args:
        out_dir (str, optional): The root directory to save checkpoints. If
            not specified, ``runner.work_dir`` will be used by default. If
            specified, the ``out_dir`` will be the concatenation of
            ``out_dir`` and the last level directory of ``runner.work_dir``.
            Defaults to None. `Changed in version 1.3.15.`
        file_client_args (dict, optional): Arguments to instantiate a
            FileClient. See :class:`mmcv.fileio.FileClient` for details.
            Defaults to None. `New in version 1.3.15.`
```

(下页继续)

(续上页)

```
    Warning:
        Before v1.3.15, the ``out_dir`` argument indicates the path where the
        checkpoint is stored. However, in v1.3.15 and later, ``out_dir``
        indicates the root directory and the final path to save checkpoint is
        the concatenation of out_dir and the last level directory of
        ``runner.work_dir``. Suppose the value of ``out_dir`` is
        "/path/of/A" and the value of ``runner.work_dir`` is "/path/of/B",
        then the final path will be "/path/of/A/B".
```

如果参数或返回值里带有需要展开描述字段的 dict，则应该采用如下格式：

```
def func(x):
    r"""
    Args:
        x (None): A dict with 2 keys, ``padded_targets``, and ``targets``.

            - ``targets`` (list[Tensor]): A list of tensors.
              Each tensor has the shape of :math:`(T_i)`. Each
              element is the index of a character.
            - ``padded_targets`` (Tensor): A tensor of shape :math:`(N)`.
              Each item is the length of a word.

    Returns:
        dict: A dict with 2 keys, ``padded_targets``, and ``targets``.

        - ``targets`` (list[Tensor]): A list of tensors.
          Each tensor has the shape of :math:`(T_i)`. Each
          element is the index of a character.
        - ``padded_targets`` (Tensor): A tensor of shape :math:`(N)`.
          Each item is the length of a word.
    """
    return x
```

重要： 为了生成 readthedocs 文档，文档的编写需要按照 ReStructrued 文档格式，否则会产生文档渲染错误，在提交 PR 前，最好生成并预览一下文档效果。语法规范参考：

- reStructuredText Primer - Sphinx documentation
- Example Google Style Python Docstrings – napoleon 0.7 documentation

17.4 注释规范

17.4.1 为什么要写注释

对于一个开源项目，团队合作以及社区之间的合作是必不可少的，因而尤其要重视合理的注释。不写注释的代码，很有可能过几个月自己也难以理解，造成额外的阅读和修改成本。

17.4.2 如何写注释

最需要写注释的是代码中那些技巧性的部分。如果你在下次代码审查的时候必须解释一下，那么你应该现在就给它写注释。对于复杂的操作，应该在其操作开始前写上若干行注释。对于不是一目了然的代码，应在其行尾添加注释。——Google 开源项目风格指南

```
# We use a weighted dictionary search to find out where i is in
# the array. We extrapolate position based on the largest num
# in the array and the array size and then do binary search to
# get the exact number.
if i & (i-1) == 0:  # True if i is 0 or a power of 2.
```

为了提高可读性, 注释应该至少离开代码 2 个空格. 另一方面, 绝不要描述代码. 假设阅读代码的人比你更懂 Python, 他只是不知道你的代码要做什么. ——Google 开源项目风格指南

```
# Wrong:
# Now go through the b array and make sure whenever i occurs
# the next element is i+1

# Wrong:
if i & (i-1) == 0:  # True if i bitwise and i-1 is 0.
```

在注释中，可以使用 Markdown 语法，因为开发人员通常熟悉 Markdown 语法，这样可以便于交流理解，如可使用单反引号表示代码和变量（注意不要和 docstring 中的 ReStructured 语法混淆）

```
# `_reversed_padding_repeated_twice` is the padding to be passed to
# `F.pad` if needed (e.g., for non-zero padding types that are
# implemented as two ops: padding + conv). `F.pad` accepts paddings in
# reverse order than the dimension.
self._reversed_padding_repeated_twice = _reverse_repeat_tuple(self.padding, 2)
```

17.4.3 注释示例

1. 出自 mmcv/utils/registry.py，对于较为复杂的逻辑结构，通过注释，明确了优先级关系。

```
# self.build_func will be set with the following priority:
# 1. build_func
# 2. parent.build_func
# 3. build_from_cfg
if build_func is None:
    if parent is not None:
        self.build_func = parent.build_func
    else:
        self.build_func = build_from_cfg
else:
    self.build_func = build_func
```

2. 出自 mmcv/runner/checkpoint.py，对于 bug 修复中的一些特殊处理，可以附带相关的 issue 链接，帮助其他人了解 bug 背景。

```
def _save_ckpt(checkpoint, file):
    # The 1.6 release of PyTorch switched torch.save to use a new
    # zipfile-based file format. It will cause RuntimeError when a
    # checkpoint was saved in high version (PyTorch version>=1.6.0) but
    # loaded in low version (PyTorch version<1.6.0). More details at
    # https://github.com/open-mmlab/mmpose/issues/904
    if digit_version(TORCH_VERSION) >= digit_version('1.6.0'):
        torch.save(checkpoint, file, _use_new_zipfile_serialization=False)
    else:
        torch.save(checkpoint, file)
```

17.5 类型注解

17.5.1 为什么要写类型注解

类型注解是对函数中变量的类型做限定或提示，为代码的安全性提供保障、增强代码的可读性、避免出现类型相关的错误。Python 没有对类型做强制限制，类型注解只起到一个提示作用，通常你的 IDE 会解析这些类型注解，然后在你调用相关代码时对类型做提示。另外也有类型注解检查工具，这些工具会根据类型注解，对代码中可能出现的问题进行检查，减少 bug 的出现。需要注意的是，通常我们不需要注释模块中的所有函数：

1. 公共的 API 需要注释
2. 在代码的安全性，清晰性和灵活性上进行权衡是否注释

3. 对于容易出现类型相关的错误的代码进行注释
4. 难以理解的代码请进行注释
5. 若代码中的类型已经稳定，可以进行注释. 对于一份成熟的代码，多数情况下，即使注释了所有的函数，也不会丧失太多的灵活性.

17.5.2 如何写类型注解

1. 函数 / 方法类型注解，通常不对 self 和 cls 注释。

```
from typing import Optional, List, Tuple

# 全部位于一行
def my_method(self, first_var: int) -> int:
    pass

# 另起一行
def my_method(
        self, first_var: int,
        second_var: float) -> Tuple[MyLongType1, MyLongType1, MyLongType1]:
    pass

# 单独成行（具体的应用场合与行宽有关，建议结合 yapf 自动化格式使用）
def my_method(
    self, first_var: int, second_var: float
) -> Tuple[MyLongType1, MyLongType1, MyLongType1]:
    pass

# 引用尚未被定义的类型
class MyClass:
    def __init__(self,
                 stack: List["MyClass"]) -> None:
        pass
```

注：类型注解中的类型可以是 Python 内置类型，也可以是自定义类，还可以使用 Python 提供的 wrapper 类对类型注解进行装饰，一些常见的注解如下：

```
# 数值类型
from numbers import Number

# 可选类型，指参数可以为 None
from typing import Optional
def foo(var: Optional[int] = None):
    pass
```

(下页继续)

(续上页)

```
# 联合类型，指同时接受多种类型
from typing import Union
def foo(var: Union[float, str]):
    pass

from typing import Sequence  # 序列类型
from typing import Iterable  # 可迭代类型
from typing import Any  # 任意类型
from typing import Callable  # 可调用类型

from typing import List, Dict  # 列表和字典的泛型类型
from typing import Tuple  # 元组的特殊格式
# 虽然在 Python 3.9 中，list, tuple 和 dict 本身已支持泛型，但为了支持之前的版本
# 我们在进行类型注解时还是需要使用 List, Tuple, Dict 类型
# 另外，在对参数类型进行注解时，尽量使用 Sequence & Iterable & Mapping
# List, Tuple, Dict 主要用于返回值类型注解
# 参见 https://docs.python.org/3/library/typing.html#typing.List
```

2. 变量类型注解，一般用于难以直接推断其类型时

```
# Recommend: 带类型注解的赋值
a: Foo = SomeUndecoratedFunction()
a: List[int]: [1, 2, 3]          # List 只支持单一类型泛型，可使用 Union
b: Tuple[int, int] = (1, 2)      # 长度固定为 2
c: Tuple[int, ...] = (1, 2, 3)  # 变长
d: Dict[str, int] = {'a': 1, 'b': 2}

# Not Recommend: 行尾类型注释
# 虽然这种方式被写在了 Google 开源指南中，但这是一种为了支持 Python 2.7 版本
# 而补充的注释方式，鉴于我们只支持 Python 3，为了风格统一，不推荐使用这种方式。
a = SomeUndecoratedFunction()  # type: Foo
a = [1, 2, 3]  # type: List[int]
b = (1, 2, 3)  # type: Tuple[int, ...]
c = (1, "2", 3.5)  # type: Tuple[int, Text, float]
```

3. 泛型

   上文中我们知道，typing 中提供了 list 和 dict 的泛型类型，那么我们自己是否可以定义类似的泛型呢?

```
from typing import TypeVar, Generic

KT = TypeVar('KT')
VT = TypeVar('VT')
```

(下页继续)

(续上页)

```
class Mapping(Generic[KT, VT]):
    def __init__(self, data: Dict[KT, VT]):
        self._data = data

    def __getitem__(self, key: KT) -> VT:
        return self._data[key]
```

使用上述方法，我们定义了一个拥有泛型能力的映射类，实际用法如下：

```
mapping = Mapping[str, float]({'a': 0.5})
value: float = example['a']
```

另外，我们也可以利用 TypeVar 在函数签名中指定联动的多个类型：

```
from typing import TypeVar, List

T = TypeVar('T')  # Can be anything
A = TypeVar('A', str, bytes)  # Must be str or bytes


def repeat(x: T, n: int) -> List[T]:
    """Return a list containing n references to x."""
    return [x]*n


def longest(x: A, y: A) -> A:
    """Return the longest of two strings."""
    return x if len(x) >= len(y) else y
```

更多关于类型注解的写法请参考 typing。

17.5.3 类型注解检查工具

mypy 是一个 Python 静态类型检查工具。根据你的类型注解，mypy 会检查传参、赋值等操作是否符合类型注解，从而避免可能出现的 bug。

例如如下的一个 Python 脚本文件 test.py:

```
def foo(var: int) -> float:
    return float(var)

a: str = foo('2.0')
b: int = foo('3.0')  # type: ignore
```

运行 mypy test.py 可以得到如下检查结果，分别指出了第 4 行在函数调用和返回值赋值两处类型错误。而第 5 行同样存在两个类型错误，由于使用了 type: ignore 而被忽略了，只有部分特殊情况可能需要此类忽略。

```
test.py:4: error: Incompatible types in assignment (expression has type "float",␣
↪variable has type "int")
test.py:4: error: Argument 1 to "foo" has incompatible type "str"; expected "int"
Found 2 errors in 1 file (checked 1 source file)
```

CHAPTER 18

mmcv.image

mmcv.image

18.1 IO

`imfrombytes`	Read an image from bytes.
`imread`	Read an image.
`imwrite`	Write image to file.
`use_backend`	Select a backend for image decoding.

18.1.1 mmcv.image.imfrombytes

mmcv.image.**imfrombytes**(*content: bytes*, *flag: str = 'color'*, *channel_order: str = 'bgr'*, *backend: Optional[str] = None*) → numpy.ndarray

Read an image from bytes.

参数

- **content** (*bytes*) –Image bytes got from files or other streams.
- **flag** (*str*) –Same as *imread()*.
- **channel_order** (*str*) –The channel order of the output, candidates are 'bgr' and 'rgb'. Default to 'bgr'.
- **backend** (*str | None*) –The image decoding backend type. Options are *cv2*, *pillow*, *turbojpeg*, *tifffile*, *None*. If backend is None, the global imread_backend specified by mmcv.use_backend() will be used. Default: None.

返回 Loaded image array.

返回类型 ndarray

实际案例

```
>>> img_path = '/path/to/img.jpg'
>>> with open(img_path, 'rb') as f:
>>>     img_buff = f.read()
>>> img = mmcv.imfrombytes(img_buff)
>>> img = mmcv.imfrombytes(img_buff, flag='color', channel_order='rgb')
>>> img = mmcv.imfrombytes(img_buff, backend='pillow')
>>> img = mmcv.imfrombytes(img_buff, backend='cv2')
```

18.1.2 mmcv.image.imread

mmcv.image.**imread**(*img_or_path: Union[numpy.ndarray, str, pathlib.Path]*, *flag: str = 'color'*, *channel_order: str = 'bgr'*, *backend: Optional[str] = None*, *file_client_args: Optional[dict] = None*, ***, *backend_args: Optional[dict] = None*) → numpy.ndarray

Read an image.

参数

- **img_or_path** (*ndarray or str or Path*) –Either a numpy array or str or pathlib.Path. If it is a numpy array (loaded image), then it will be returned as is.
- **flag** (*str*) –Flags specifying the color type of a loaded image, candidates are *color*, *grayscale*, *unchanged*, *color_ignore_orientation* and *grayscale_ignore_orientation*. By default,

cv2 and *pillow* backend would rotate the image according to its EXIF info unless called with *unchanged* or **_ignore_orientation* flags. *turbojpeg* and *tifffile* backend always ignore image's EXIF info regardless of the flag. The *turbojpeg* backend only supports *color* and *grayscale*.

- **channel_order** (*str*) –Order of channel, candidates are *bgr* and *rgb*.
- **backend** (*str | None*) –The image decoding backend type. Options are *cv2*, *pillow*, *turbojpeg*, *tifffile*, *None*. If backend is None, the global imread_backend specified by `mmcv.use_backend()` will be used. Default: None.
- **file_client_args** (*dict, optional*) –Arguments to instantiate a FileClient. See `mmengine.fileio.FileClient` for details. Default: None. It will be deprecated in future. Please use `backend_args` instead. Deprecated in version 2.0.0rc4.
- **backend_args** (*dict, optional*) –Instantiates the corresponding file backend. It may contain *backend* key to specify the file backend. If it contains, the file backend corresponding to this value will be used and initialized with the remaining values, otherwise the corresponding file backend will be selected based on the prefix of the file path. Defaults to None. New in version 2.0.0rc4.

返回 Loaded image array.

返回类型 ndarray

实际案例

```
>>> import mmcv
>>> img_path = '/path/to/img.jpg'
>>> img = mmcv.imread(img_path)
>>> img = mmcv.imread(img_path, flag='color', channel_order='rgb',
...     backend='cv2')
>>> img = mmcv.imread(img_path, flag='color', channel_order='bgr',
...     backend='pillow')
>>> s3_img_path = 's3://bucket/img.jpg'
>>> # infer the file backend by the prefix s3
>>> img = mmcv.imread(s3_img_path)
>>> # manually set the file backend petrel
>>> img = mmcv.imread(s3_img_path, backend_args={
...     'backend': 'petrel'})
>>> http_img_path = 'http://path/to/img.jpg'
>>> img = mmcv.imread(http_img_path)
>>> img = mmcv.imread(http_img_path, backend_args={
...     'backend': 'http'})
```

18.1.3 mmcv.image.imwrite

mmcv.image.**imwrite**(*img: numpy.ndarray*, *file_path: str*, *params: Optional[list] = None*, *auto_mkdir: Optional[bool] = None*, *file_client_args: Optional[dict] = None*, ***, *backend_args: Optional[dict] = None*) → bool

Write image to file.

> **警告:** The parameter *auto_mkdir* will be deprecated in the future and every file clients will make directory automatically.

参数

- **img** (*ndarray*) –Image array to be written.
- **file_path** (*str*) –Image file path.
- **params** (*None or list*) –Same as opencv *imwrite()* interface.
- **auto_mkdir** (*bool*) –If the parent folder of *file_path* does not exist, whether to create it automatically. It will be deprecated.
- **file_client_args** (*dict, optional*) –Arguments to instantiate a FileClient. See mmengine.fileio.FileClient for details. Default: None. It will be deprecated in future. Please use backend_args instead. Deprecated in version 2.0.0rc4.
- **backend_args** (*dict, optional*) –Instantiates the corresponding file backend. It may contain *backend* key to specify the file backend. If it contains, the file backend corresponding to this value will be used and initialized with the remaining values, otherwise the corresponding file backend will be selected based on the prefix of the file path. Defaults to None. New in version 2.0.0rc4.

返回 Successful or not.

返回类型 bool

实际案例

```
>>> # write to hard disk client
>>> ret = mmcv.imwrite(img, '/path/to/img.jpg')
>>> # infer the file backend by the prefix s3
>>> ret = mmcv.imwrite(img, 's3://bucket/img.jpg')
>>> # manually set the file backend petrel
>>> ret = mmcv.imwrite(img, 's3://bucket/img.jpg', backend_args={
...     'backend': 'petrel'})
```

18.1.4 mmcv.image.use_backend

mmcv.image.**use_backend**(*backend: str*) → None

Select a backend for image decoding.

参数

- **backend** (*str*) –The image decoding backend type. Options are *cv2*,
- **pillow** –//github.com/lilohuang/PyTurboJPEG)
- **(see https** (*turbojpeg*) –//github.com/lilohuang/PyTurboJPEG)
- **tifffile. turbojpeg is faster but it only supports .jpeg** (*and*) –
- **format.** (*file*) –

18.2 Color Space

bgr2gray	Convert a BGR image to grayscale image.
bgr2hls	Convert a BGR image to HLS
bgr2hsv	Convert a BGR image to HSV
bgr2rgb	Convert a BGR image to RGB
bgr2ycbcr	Convert a BGR image to YCbCr image.
gray2bgr	Convert a grayscale image to BGR image.
gray2rgb	Convert a grayscale image to RGB image.
hls2bgr	Convert a HLS image to BGR
hsv2bgr	Convert a HSV image to BGR
imconvert	Convert an image from the src colorspace to dst colorspace.
rgb2bgr	Convert a RGB image to BGR
rgb2gray	Convert a RGB image to grayscale image.
rgb2ycbcr	Convert a RGB image to YCbCr image.
ycbcr2bgr	Convert a YCbCr image to BGR image.
ycbcr2rgb	Convert a YCbCr image to RGB image.

18.2.1 mmcv.image.bgr2gray

mmcv.image.**bgr2gray**(*img: numpy.ndarray*, *keepdim: bool = False*) → numpy.ndarray

Convert a BGR image to grayscale image.

参数

- **img** (*ndarray*) –The input image.
- **keepdim** (*bool*) –If False (by default), then return the grayscale image with 2 dims, otherwise 3 dims.

返回 The converted grayscale image.

返回类型 ndarray

18.2.2 mmcv.image.bgr2hls

mmcv.image.**bgr2hls**(*img: numpy.ndarray*) → numpy.ndarray

Convert a BGR image to HLS image.

参数 **img** (*ndarray or str*) –The input image.

返回 The converted HLS image.

返回类型 ndarray

18.2.3 mmcv.image.bgr2hsv

mmcv.image.**bgr2hsv**(*img: numpy.ndarray*) → numpy.ndarray

Convert a BGR image to HSV image.

参数 **img** (*ndarray or str*) –The input image.

返回 The converted HSV image.

返回类型 ndarray

18.2.4 mmcv.image.bgr2rgb

mmcv.image.**bgr2rgb**(*img: numpy.ndarray*) → numpy.ndarray

Convert a BGR image to RGB image.

参数 **img** (*ndarray or str*) –The input image.

返回 The converted RGB image.

返回类型 ndarray

18.2.5 mmcv.image.bgr2ycbcr

mmcv.image.**bgr2ycbcr**(*img: numpy.ndarray*, *y_only: bool = False*) → numpy.ndarray

Convert a BGR image to YCbCr image.

The bgr version of rgb2ycbcr. It implements the ITU-R BT.601 conversion for standard-definition television. See more details in https://en.wikipedia.org/wiki/YCbCr#ITU-R_BT.601_conversion.

It differs from a similar function in cv2.cvtColor: *BGR <-> YCrCb*. In OpenCV, it implements a JPEG conversion. See more details in https://en.wikipedia.org/wiki/YCbCr#JPEG_conversion.

参数

- **img** (*ndarray*) –The input image. It accepts: 1. np.uint8 type with range [0, 255]; 2. np.float32 type with range [0, 1].
- **y_only** (*bool*) –Whether to only return Y channel. Default: False.

返回 The converted YCbCr image. The output image has the same type and range as input image.

返回类型 ndarray

18.2.6 mmcv.image.gray2bgr

mmcv.image.**gray2bgr**(*img: numpy.ndarray*) → numpy.ndarray

Convert a grayscale image to BGR image.

参数 **img** (*ndarray*) –The input image.

返回 The converted BGR image.

返回类型 ndarray

18.2.7 mmcv.image.gray2rgb

mmcv.image.**gray2rgb**(*img: numpy.ndarray*) → numpy.ndarray

Convert a grayscale image to RGB image.

参数 **img** (*ndarray*) –The input image.

返回 The converted RGB image.

返回类型 ndarray

18.2.8 mmcv.image.hls2bgr

mmcv.image.**hls2bgr**(*img: numpy.ndarray*) → numpy.ndarray

Convert a HLS image to BGR image.

参数 **img** (*ndarray or str*) –The input image.

返回 The converted BGR image.

返回类型 ndarray

18.2.9 mmcv.image.hsv2bgr

mmcv.image.**hsv2bgr**(*img: numpy.ndarray*) → numpy.ndarray

Convert a HSV image to BGR image.

参数 **img** (*ndarray or str*) –The input image.

返回 The converted BGR image.

返回类型 ndarray

18.2.10 mmcv.image.imconvert

mmcv.image.**imconvert**(*img: numpy.ndarray*, *src: str*, *dst: str*) → numpy.ndarray

Convert an image from the src colorspace to dst colorspace.

参数

- **img** (*ndarray*) –The input image.
- **src** (*str*) –The source colorspace, e.g., ‘rgb’, ‘hsv’.
- **dst** (*str*) –The destination colorspace, e.g., ‘rgb’, ‘hsv’.

返回 The converted image.

返回类型 ndarray

18.2.11 mmcv.image.rgb2bgr

mmcv.image.**rgb2bgr**(*img: numpy.ndarray*) → numpy.ndarray

Convert a RGB image to BGR image.

参数 **img** (*ndarray or str*) –The input image.

返回 The converted BGR image.

返回类型 ndarray

18.2.12 mmcv.image.rgb2gray

mmcv.image.**rgb2gray**(*img: numpy.ndarray*, *keepdim: bool = False*) → numpy.ndarray

Convert a RGB image to grayscale image.

参数

- **img** (*ndarray*) –The input image.
- **keepdim** (*bool*) –If False (by default), then return the grayscale image with 2 dims, otherwise 3 dims.

返回 The converted grayscale image.

返回类型 ndarray

18.2.13 mmcv.image.rgb2ycbcr

mmcv.image.**rgb2ycbcr**(*img: numpy.ndarray*, *y_only: bool = False*) → numpy.ndarray

Convert a RGB image to YCbCr image.

This function produces the same results as Matlab's *rgb2ycbcr* function. It implements the ITU-R BT.601 conversion for standard-definition television. See more details in https://en.wikipedia.org/wiki/YCbCr#ITU-R_BT.601_conversion.

It differs from a similar function in cv2.cvtColor: *RGB <-> YCrCb*. In OpenCV, it implements a JPEG conversion. See more details in https://en.wikipedia.org/wiki/YCbCr#JPEG_conversion.

参数

- **img** (*ndarray*) –The input image. It accepts: 1. np.uint8 type with range [0, 255]; 2. np.float32 type with range [0, 1].

- **y_only** (*bool*) –Whether to only return Y channel. Default: False.

返回 The converted YCbCr image. The output image has the same type and range as input image.

返回类型 ndarray

18.2.14 mmcv.image.ycbcr2bgr

mmcv.image.**ycbcr2bgr**(*img: numpy.ndarray*) → numpy.ndarray

Convert a YCbCr image to BGR image.

The bgr version of ycbcr2rgb. It implements the ITU-R BT.601 conversion for standard-definition television. See more details in https://en.wikipedia.org/wiki/YCbCr#ITU-R_BT.601_conversion.

It differs from a similar function in cv2.cvtColor: *YCrCb <-> BGR*. In OpenCV, it implements a JPEG conversion. See more details in https://en.wikipedia.org/wiki/YCbCr#JPEG_conversion.

参数 **img** (*ndarray*) –The input image. It accepts: 1. np.uint8 type with range [0, 255]; 2. np.float32 type with range [0, 1].

返回 The converted BGR image. The output image has the same type and range as input image.

返回类型 ndarray

18.2.15 mmcv.image.ycbcr2rgb

mmcv.image.**ycbcr2rgb**(*img: numpy.ndarray*) → numpy.ndarray

Convert a YCbCr image to RGB image.

This function produces the same results as Matlab's ycbcr2rgb function. It implements the ITU-R BT.601 conversion for standard-definition television. See more details in https://en.wikipedia.org/wiki/YCbCr#ITU-R_BT.601_conversion.

It differs from a similar function in cv2.cvtColor: *YCrCb <-> RGB*. In OpenCV, it implements a JPEG conversion. See more details in https://en.wikipedia.org/wiki/YCbCr#JPEG_conversion.

参数 **img** (*ndarray*) –The input image. It accepts: 1. np.uint8 type with range [0, 255]; 2. np.float32 type with range [0, 1].

返回 The converted RGB image. The output image has the same type and range as input image.

返回类型 ndarray

18.3 Geometric

cutout	Randomly cut out a rectangle from the original img.
imcrop	Crop image patches.
imflip	Flip an image horizontally or vertically.
impad	Pad the given image to a certain shape or pad on all sides with specified padding mode and padding value.
impad_to_multiple	Pad an image to ensure each edge to be multiple to some number.
imrescale	Resize image while keeping the aspect ratio.
imresize	Resize image to a given size.
imresize_like	Resize image to the same size of a given image.
imresize_to_multiple	Resize image according to a given size or scale factor and then rounds up the the resized or rescaled image size to the nearest value that can be divided by the divisor.
imrotate	Rotate an image.
imshear	Shear an image.
imtranslate	Translate an image.
rescale_size	Calculate the new size to be rescaled to.

18.3.1 mmcv.image.cutout

mmcv.image.**cutout**(*img: numpy.ndarray*, *shape: Union[int, Tuple[int, int]]*, *pad_val: Union[int, float, tuple] = 0*) → numpy.ndarray

Randomly cut out a rectangle from the original img.

参数

- **img** (*ndarray*) –Image to be cutout.
- **shape** (*int | tuple[int]*) –Expected cutout shape (h, w). If given as a int, the value will be used for both h and w.
- **pad_val** (*int | float | tuple[int | float]*) –Values to be filled in the cut area. Defaults to 0.

返回 The cutout image.

返回类型 ndarray

18.3.2 mmcv.image.imcrop

mmcv.image.**imcrop**(*img: numpy.ndarray*, *bboxes: numpy.ndarray*, *scale: float = 1.0*, *pad_fill: Optional[Union[float, list]] = None*) → Union[numpy.ndarray, List[numpy.ndarray]]

Crop image patches.

3 steps: scale the bboxes -> clip bboxes -> crop and pad.

参数

- **img** (*ndarray*) –Image to be cropped.
- **bboxes** (*ndarray*) –Shape (k, 4) or (4,), location of cropped bboxes.
- **scale** (*float, optional*) –Scale ratio of bboxes, the default value 1.0 means no scaling.
- **pad_fill** (*Number | list[Number]*) –Value to be filled for padding. Default: None, which means no padding.

返回 The cropped image patches.

返回类型 list[ndarray] | ndarray

18.3.3 mmcv.image.imflip

mmcv.image.**imflip**(*img: numpy.ndarray*, *direction: str = 'horizontal'*) → numpy.ndarray

Flip an image horizontally or vertically.

参数

- **img** (*ndarray*) –Image to be flipped.
- **direction** (*str*) –The flip direction, either "horizontal" or "vertical" or "diagonal".

返回 The flipped image.

返回类型 ndarray

18.3.4 mmcv.image.impad

mmcv.image.**impad**(*img: numpy.ndarray*, *, *shape: Optional[Tuple[int, int]] = None*, *padding: Optional[Union[int, tuple]] = None*, *pad_val: Union[float, List] = 0*, *padding_mode: str = 'constant'*) → numpy.ndarray

Pad the given image to a certain shape or pad on all sides with specified padding mode and padding value.

参数

- **img** (*ndarray*) –Image to be padded.
- **shape** (*tuple[int]*) –Expected padding shape (h, w). Default: None.

- **padding** (*int or tuple[int]*) –Padding on each border. If a single int is provided this is used to pad all borders. If tuple of length 2 is provided this is the padding on left/right and top/bottom respectively. If a tuple of length 4 is provided this is the padding for the left, top, right and bottom borders respectively. Default: None. Note that *shape* and *padding* can not be both set.
- **pad_val** (*Number | Sequence[Number]*) –Values to be filled in padding areas when padding_mode is 'constant'. Default: 0.
- **padding_mode** (*str*) –Type of padding. Should be: constant, edge, reflect or symmetric. Default: constant.
  - constant: pads with a constant value, this value is specified with pad_val.
  - edge: pads with the last value at the edge of the image.
  - reflect: pads with reflection of image without repeating the last value on the edge. For example, padding [1, 2, 3, 4] with 2 elements on both sides in reflect mode will result in [3, 2, 1, 2, 3, 4, 3, 2].
  - symmetric: pads with reflection of image repeating the last value on the edge. For example, padding [1, 2, 3, 4] with 2 elements on both sides in symmetric mode will result in [2, 1, 1, 2, 3, 4, 4, 3]

返回 The padded image.

返回类型 ndarray

18.3.5 mmcv.image.impad_to_multiple

mmcv.image.**impad_to_multiple**(*img: numpy.ndarray*, *divisor: int*, *pad_val: Union[float, List] = 0*) → numpy.ndarray

Pad an image to ensure each edge to be multiple to some number.

参数

- **img** (*ndarray*) –Image to be padded.
- **divisor** (*int*) –Padded image edges will be multiple to divisor.
- **pad_val** (*Number | Sequence[Number]*) –Same as *impad()*.

返回 The padded image.

返回类型 ndarray

18.3.6 mmcv.image.imrescale

mmcv.image.**imrescale**(*img: numpy.ndarray*, *scale: Union[float, Tuple[int, int]]*, *return_scale: bool = False*, *interpolation: str = 'bilinear'*, *backend: Optional[str] = None*) → Union[numpy.ndarray, Tuple[numpy.ndarray, float]]

Resize image while keeping the aspect ratio.

参数

- **img** (*ndarray*) –The input image.
- **scale** (*float | tuple[int]*) –The scaling factor or maximum size. If it is a float number, then the image will be rescaled by this factor, else if it is a tuple of 2 integers, then the image will be rescaled as large as possible within the scale.
- **return_scale** (*bool*) –Whether to return the scaling factor besides the rescaled image.
- **interpolation** (*str*) –Same as `resize()`.
- **backend** (*str | None*) –Same as `resize()`.

返回 The rescaled image.

返回类型 ndarray

18.3.7 mmcv.image.imresize

mmcv.image.**imresize**(*img: numpy.ndarray*, *size: Tuple[int, int]*, *return_scale: bool = False*, *interpolation: str = 'bilinear'*, *out: Optional[numpy.ndarray] = None*, *backend: Optional[str] = None*) → Union[Tuple[numpy.ndarray, float, float], numpy.ndarray]

Resize image to a given size.

参数

- **img** (*ndarray*) –The input image.
- **size** (*tuple[int]*) –Target size (w, h).
- **return_scale** (*bool*) –Whether to return *w_scale* and *h_scale*.
- **interpolation** (*str*) –Interpolation method, accepted values are "nearest", "bilinear", "bicubic", "area", "lanczos" for 'cv2' backend, "nearest", "bilinear" for 'pillow' backend.
- **out** (*ndarray*) –The output destination.
- **backend** (*str | None*) –The image resize backend type. Options are *cv2*, *pillow*, *None*. If backend is None, the global imread_backend specified by `mmcv.use_backend()` will be used. Default: None.

返回 (*resized_img*, *w_scale*, *h_scale*) or *resized_img*.

返回类型 tuple | ndarray

18.3.8 mmcv.image.imresize_like

mmcv.image.**imresize_like**(*img: numpy.ndarray*, *dst_img: numpy.ndarray*, *return_scale: bool = False*, *interpolation: str = 'bilinear'*, *backend: Optional[str] = None*) → Union[Tuple[numpy.ndarray, float, float], numpy.ndarray]

Resize image to the same size of a given image.

参数

- **img** (*ndarray*) –The input image.
- **dst_img** (*ndarray*) –The target image.
- **return_scale** (*bool*) –Whether to return *w_scale* and *h_scale*.
- **interpolation** (*str*) –Same as `resize()`.
- **backend** (*str | None*) –Same as `resize()`.

返回 (*resized_img*, *w_scale*, *h_scale*) or *resized_img*.

返回类型 tuple or ndarray

18.3.9 mmcv.image.imresize_to_multiple

mmcv.image.**imresize_to_multiple**(*img: numpy.ndarray*, *divisor: Union[int, Tuple[int, int]]*, *size: Optional[Union[int, Tuple[int, int]]] = None*, *scale_factor: Optional[Union[float, Tuple[float, float]]] = None*, *keep_ratio: bool = False*, *return_scale: bool = False*, *interpolation: str = 'bilinear'*, *out: Optional[numpy.ndarray] = None*, *backend: Optional[str] = None*) → Union[Tuple[numpy.ndarray, float, float], numpy.ndarray]

Resize image according to a given size or scale factor and then rounds up the the resized or rescaled image size to the nearest value that can be divided by the divisor.

参数

- **img** (*ndarray*) –The input image.
- **divisor** (*int | tuple*) –Resized image size will be a multiple of divisor. If divisor is a tuple, divisor should be (w_divisor, h_divisor).
- **size** (*None | int | tuple[int]*) –Target size (w, h). Default: None.
- **scale_factor** (*None | float | tuple[float]*) –Multiplier for spatial size. Should match input size if it is a tuple and the 2D style is (w_scale_factor, h_scale_factor). Default: None.

- **keep_ratio** (*bool*) –Whether to keep the aspect ratio when resizing the image. Default: False.
- **return_scale** (*bool*) –Whether to return *w_scale* and *h_scale*.
- **interpolation** (*str*) –Interpolation method, accepted values are "nearest", "bilinear", "bicubic", "area", "lanczos" for 'cv2' backend, "nearest", "bilinear" for 'pillow' backend.
- **out** (*ndarray*) –The output destination.
- **backend** (*str | None*) –The image resize backend type. Options are *cv2*, *pillow*, *None*. If backend is None, the global imread_backend specified by `mmcv.use_backend()` will be used. Default: None.

返回 (*resized_img*, *w_scale*, *h_scale*) or *resized_img*.

返回类型 tuple | ndarray

18.3.10 mmcv.image.imrotate

mmcv.image.**imrotate**(*img: numpy.ndarray*, *angle: float*, *center: Optional[Tuple[float, float]] = None*, *scale: float = 1.0*, *border_value: int = 0*, *interpolation: str = 'bilinear'*, *auto_bound: bool = False*, *border_mode: str = 'constant'*) → numpy.ndarray

Rotate an image.

参数

- **img** (*np.ndarray*) –Image to be rotated.
- **angle** (*float*) –Rotation angle in degrees, positive values mean clockwise rotation.
- **center** (*tuple[float], optional*) –Center point (w, h) of the rotation in the source image. If not specified, the center of the image will be used.
- **scale** (*float*) –Isotropic scale factor.
- **border_value** (*int*) –Border value used in case of a constant border. Defaults to 0.
- **interpolation** (*str*) –Same as `resize()`.
- **auto_bound** (*bool*) –Whether to adjust the image size to cover the whole rotated image.
- **border_mode** (*str*) –Pixel extrapolation method. Defaults to 'constant'.

返回 The rotated image.

返回类型 np.ndarray

18.3.11 mmcv.image.imshear

mmcv.image.**imshear**(*img: numpy.ndarray*, *magnitude: Union[int, float]*, *direction: str = 'horizontal'*, *border_value: Union[int, Tuple[int, int]] = 0*, *interpolation: str = 'bilinear'*) → numpy.ndarray

Shear an image.

参数

- **img** (*ndarray*) –Image to be sheared with format (h, w) or (h, w, c).
- **magnitude** (*int | float*) –The magnitude used for shear.
- **direction** (*str*) –The flip direction, either "horizontal" or "vertical".
- **border_value** (*int | tuple[int]*) –Value used in case of a constant border.
- **interpolation** (*str*) –Same as `resize()`.

返回 The sheared image.

返回类型 ndarray

18.3.12 mmcv.image.imtranslate

mmcv.image.**imtranslate**(*img: numpy.ndarray*, *offset: Union[int, float]*, *direction: str = 'horizontal'*, *border_value: Union[int, tuple] = 0*, *interpolation: str = 'bilinear'*) → numpy.ndarray

Translate an image.

参数

- **img** (*ndarray*) –Image to be translated with format (h, w) or (h, w, c).
- **offset** (*int | float*) –The offset used for translate.
- **direction** (*str*) –The translate direction, either "horizontal" or "vertical".
- **border_value** (*int | tuple[int]*) –Value used in case of a constant border.
- **interpolation** (*str*) –Same as `resize()`.

返回 The translated image.

返回类型 ndarray

18.3.13 mmcv.image.rescale_size

mmcv.image.**rescale_size**(*old_size: tuple*, *scale: Union[float, int, tuple]*, *return_scale: bool = False*) → tuple

Calculate the new size to be rescaled to.

参数

- **old_size** (*tuple[int]*) –The old size (w, h) of image.
- **scale** (*float | tuple[int]*) –The scaling factor or maximum size. If it is a float number, then the image will be rescaled by this factor, else if it is a tuple of 2 integers, then the image will be rescaled as large as possible within the scale.
- **return_scale** (*bool*) –Whether to return the scaling factor besides the rescaled image size.

返回 The new rescaled image size.

返回类型 tuple[int]

18.4 Photometric

adjust_brightness	Adjust image brightness.
adjust_color	It blends the source image and its gray image:
adjust_contrast	Adjust image contrast.
adjust_hue	Adjust hue of an image.
adjust_lighting	AlexNet-style PCA jitter.
adjust_sharpness	Adjust image sharpness.
auto_contrast	Auto adjust image contrast.
clahe	Use CLAHE method to process the image.
imdenormalize	
imequalize	Equalize the image histogram.
iminvert	Invert (negate) an image.
imnormalize	Normalize an image with mean and std.
lut_transform	Transform array by look-up table.
posterize	Posterize an image (reduce the number of bits for each color channel)
solarize	Solarize an image (invert all pixel values above a threshold)

18.4.1 mmcv.image.adjust_brightness

mmcv.image.**adjust_brightness**(*img*, *factor=1.0*, *backend=None*)

Adjust image brightness.

This function controls the brightness of an image. An enhancement factor of 0.0 gives a black image. A factor of 1.0 gives the original image. This function blends the source image and the degenerated black image:

$$output = img * factor + degenerated * (1 - factor)$$

参数

- **img** (*ndarray*) –Image to be brightened.
- **factor** (*float*) –A value controls the enhancement. Factor 1.0 returns the original image, lower factors mean less color (brightness, contrast, etc), and higher values more. Default 1.
- **backend** (*str | None*) –The image processing backend type. Options are *cv2*, *pillow*, *None*. If backend is None, the global `imread_backend` specified by `mmcv.use_backend()` will be used. Defaults to None.

返回 The brightened image.

返回类型 ndarray

18.4.2 mmcv.image.adjust_color

mmcv.image.**adjust_color**(*img*, *alpha=1*, *beta=None*, *gamma=0*, *backend=None*)

It blends the source image and its gray image:

$$output = img * alpha + gray\_img * beta + gamma$$

参数

- **img** (*ndarray*) –The input source image.
- **alpha** (*int | float*) –Weight for the source image. Default 1.
- **beta** (*int | float*) –Weight for the converted gray image. If None, it's assigned the value (1 - *alpha*).
- **gamma** (*int | float*) –Scalar added to each sum. Same as `cv2.addWeighted()`. Default 0.
- **backend** (*str | None*) –The image processing backend type. Options are *cv2*, *pillow*, *None*. If backend is None, the global `imread_backend` specified by `mmcv.use_backend()` will be used. Defaults to None.

返回 Colored image which has the same size and dtype as input.

返回类型 ndarray

18.4.3 mmcv.image.adjust_contrast

mmcv.image.**adjust_contrast**(*img*, *factor=1.0*, *backend=None*)

Adjust image contrast.

This function controls the contrast of an image. An enhancement factor of 0.0 gives a solid grey image. A factor of 1.0 gives the original image. It blends the source image and the degenerated mean image:

$$output = img * factor + degenerated * (1 - factor)$$

参数

- **img** (*ndarray*) –Image to be contrasted. BGR order.
- **factor** (*float*) –Same as mmcv.adjust_brightness().
- **backend** (*str | None*) –The image processing backend type. Options are *cv2*, *pillow*, *None*. If backend is None, the global imread_backend specified by mmcv.use_backend() will be used. Defaults to None.

返回 The contrasted image.

返回类型 ndarray

18.4.4 mmcv.image.adjust_hue

mmcv.image.**adjust_hue**(*img: numpy.ndarray*, *hue_factor: float*, *backend: Optional[str] = None*) → numpy.ndarray

Adjust hue of an image.

The image hue is adjusted by converting the image to HSV and cyclically shifting the intensities in the hue channel (H). The image is then converted back to original image mode.

hue_factor is the amount of shift in H channel and must be in the interval *[-0.5, 0.5]*.

Modified from https://github.com/pytorch/vision/blob/main/torchvision/ transforms/functional.py

参数

- **img** (*ndarray*) –Image to be adjusted.
- **hue_factor** (*float*) –How much to shift the hue channel. Should be in [-0.5, 0.5]. 0.5 and -0.5 give complete reversal of hue channel in HSV space in positive and negative direction respectively. 0 means no shift. Therefore, both -0.5 and 0.5 will give an image with complementary colors while 0 gives the original image.
- **backend** (*str | None*) –The image processing backend type. Options are *cv2*, *pillow*, *None*. If backend is None, the global imread_backend specified by mmcv.use_backend() will be used. Defaults to None.

返回 Hue adjusted image.

返回类型 ndarray

18.4.5 mmcv.image.adjust_lighting

mmcv.image.**adjust_lighting**(*img*, *eigval*, *eigvec*, *alphastd=0.1*, *to_rgb=True*)

AlexNet-style PCA jitter.

This data augmentation is proposed in ImageNet Classification with Deep Convolutional Neural Networks.

参数

- **img** (*ndarray*) –Image to be adjusted lighting. BGR order.
- **eigval** (*ndarray*) –the eigenvalue of the convariance matrix of pixel values, respectively.
- **eigvec** (*ndarray*) –the eigenvector of the convariance matrix of pixel values, respectively.
- **alphastd** (*float*) –The standard deviation for distribution of alpha. Defaults to 0.1
- **to_rgb** (*bool*) –Whether to convert img to rgb.

返回 The adjusted image.

返回类型 ndarray

18.4.6 mmcv.image.adjust_sharpness

mmcv.image.**adjust_sharpness**(*img*, *factor=1.0*, *kernel=None*)

Adjust image sharpness.

This function controls the sharpness of an image. An enhancement factor of 0.0 gives a blurred image. A factor of 1.0 gives the original image. And a factor of 2.0 gives a sharpened image. It blends the source image and the degenerated mean image:

$$output = img * factor + degenerated * (1 - factor)$$

参数

- **img** (*ndarray*) –Image to be sharpened. BGR order.
- **factor** (*float*) –Same as `mmcv.adjust_brightness()`.
- **kernel** (*np.ndarray, optional*) –Filter kernel to be applied on the img to obtain the degenerated img. Defaults to None.

注解: No value sanity check is enforced on the kernel set by users. So with an inappropriate kernel, the `adjust_sharpness` may fail to perform the function its name indicates but end up performing whatever transform determined by the kernel.

返回 The sharpened image.

返回类型 ndarray

18.4.7 mmcv.image.auto_contrast

mmcv.image.**auto_contrast**(*img*, *cutoff=0*)

Auto adjust image contrast.

This function maximize (normalize) image contrast by first removing cutoff percent of the lightest and darkest pixels from the histogram and remapping the image so that the darkest pixel becomes black (0), and the lightest becomes white (255).

参数

- **img** (*ndarray*) –Image to be contrasted. BGR order.
- **cutoff** (*int | float | tuple*) –The cutoff percent of the lightest and darkest pixels to be removed. If given as tuple, it shall be (low, high). Otherwise, the single value will be used for both. Defaults to 0.

返回 The contrasted image.

返回类型 ndarray

18.4.8 mmcv.image.clahe

mmcv.image.**clahe**(*img*, *clip_limit=40.0*, *tile_grid_size=(8, 8)*)

Use CLAHE method to process the image.

See *ZUIDERVELD,K. Contrast Limited Adaptive Histogram Equalization[J]. Graphics Gems, 1994:474-485.* for more information.

参数

- **img** (*ndarray*) –Image to be processed.
- **clip_limit** (*float*) –Threshold for contrast limiting. Default: 40.0.
- **tile_grid_size** (*tuple[int]*) –Size of grid for histogram equalization. Input image will be divided into equally sized rectangular tiles. It defines the number of tiles in row and column. Default: (8, 8).

返回 The processed image.

返回类型 ndarray

18.4.9 mmcv.image.imdenormalize

mmcv.image.**imdenormalize**(*img*, *mean*, *std*, *to_bgr=True*)

18.4.10 mmcv.image.imequalize

mmcv.image.**imequalize**(*img*)

Equalize the image histogram.

This function applies a non-linear mapping to the input image, in order to create a uniform distribution of grayscale values in the output image.

参数 **img** (*ndarray*) –Image to be equalized.

返回 The equalized image.

返回类型 ndarray

18.4.11 mmcv.image.iminvert

mmcv.image.**iminvert**(*img*)

Invert (negate) an image.

参数 **img** (*ndarray*) –Image to be inverted.

返回 The inverted image.

返回类型 ndarray

18.4.12 mmcv.image.imnormalize

mmcv.image.**imnormalize**(*img*, *mean*, *std*, *to_rgb=True*)

Normalize an image with mean and std.

参数

- **img** (*ndarray*) –Image to be normalized.
- **mean** (*ndarray*) –The mean to be used for normalize.
- **std** (*ndarray*) –The std to be used for normalize.
- **to_rgb** (*bool*) –Whether to convert to rgb.

返回 The normalized image.

返回类型 ndarray

18.4.13 mmcv.image.lut_transform

mmcv.image.**lut_transform**(*img*, *lut_table*)

Transform array by look-up table.

The function lut_transform fills the output array with values from the look-up table. Indices of the entries are taken from the input array.

参数

- **img** (*ndarray*) –Image to be transformed.
- **lut_table** (*ndarray*) –look-up table of 256 elements; in case of multi-channel input array, the table should either have a single channel (in this case the same table is used for all channels) or the same number of channels as in the input array.

返回 The transformed image.

返回类型 ndarray

18.4.14 mmcv.image.posterize

mmcv.image.**posterize**(*img*, *bits*)

Posterize an image (reduce the number of bits for each color channel)

参数

- **img** (*ndarray*) –Image to be posterized.
- **bits** (*int*) –Number of bits (1 to 8) to use for posterizing.

返回 The posterized image.

返回类型 ndarray

18.4.15 mmcv.image.solarize

mmcv.image.**solarize**(*img*, *thr=128*)

Solarize an image (invert all pixel values above a threshold)

参数

- **img** (*ndarray*) –Image to be solarized.
- **thr** (*int*) –Threshold for solarizing (0 - 255).

返回 The solarized image.

返回类型 ndarray

18.5 Miscellaneous

tensor2imgs	Convert tensor to 3-channel images or 1-channel gray images.

18.5.1 mmcv.image.tensor2imgs

mmcv.image.**tensor2imgs**(*tensor*, *mean: Optional[tuple] = None*, *std: Optional[tuple] = None*, *to_rgb: bool = True*) → list

Convert tensor to 3-channel images or 1-channel gray images.

参数

- **tensor** (*torch.Tensor*) –Tensor that contains multiple images, shape (N, C, H, W). *C* can be either 3 or 1.
- **mean** (*tuple[float], optional*) –Mean of images. If None, (0, 0, 0) will be used for tensor with 3-channel, while (0,) for tensor with 1-channel. Defaults to None.
- **std** (*tuple[float], optional*) –Standard deviation of images. If None, (1, 1, 1) will be used for tensor with 3-channel, while (1,) for tensor with 1-channel. Defaults to None.
- **to_rgb** (*bool, optional*) –Whether the tensor was converted to RGB format in the first place. If so, convert it back to BGR. For the tensor with 1 channel, it must be False. Defaults to True.

返回 A list that contains multiple images.

返回类型 list[np.ndarray]

CHAPTER 19

mmcv.video

mmcv.video

19.1 IO

`VideoReader`	Video class with similar usage to a list object.
`Cache`	

19.1.1 VideoReader

class mmcv.video.**VideoReader**(*filename*, *cache_capacity=10*)

Video class with similar usage to a list object.

This video wrapper class provides convenient apis to access frames. There exists an issue of OpenCV's VideoCapture class that jumping to a certain frame may be inaccurate. It is fixed in this class by checking the position after jumping each time. Cache is used when decoding videos. So if the same frame is visited for the second time, there is no need to decode again if it is stored in the cache.

实际案例

```
>>> import mmcv
>>> v = mmcv.VideoReader('sample.mp4')
>>> len(v)  # get the total frame number with `len()`
120
>>> for img in v:  # v is iterable
>>>     mmcv.imshow(img)
>>> v[5]  # get the 6th frame
```

current_frame()

Get the current frame (frame that is just visited).

返回 If the video is fresh, return None, otherwise return the frame.

返回类型 ndarray or None

cvt2frames(*frame_dir*, *file_start=0*, *filename_tmpl='{:06d}.jpg'*, *start=0*, *max_num=0*, *show_progress=True*)

Convert a video to frame images.

参数

- **frame_dir** (*str*) –Output directory to store all the frame images.
- **file_start** (*int*) –Filenames will start from the specified number.
- **filename_tmpl** (*str*) –Filename template with the index as the placeholder.
- **start** (*int*) –The starting frame index.
- **max_num** (*int*) –Maximum number of frames to be written.
- **show_progress** (*bool*) –Whether to show a progress bar.

property fourcc

"Four character code" of the video.

Type str

property fps

FPS of the video.

Type float

property frame_cnt

Total frames of the video.

Type int

get_frame(*frame_id*)

Get frame by index.

参数 **frame_id** (*int*) –Index of the expected frame, 0-based.

返回 Return the frame if successful, otherwise None.

返回类型 ndarray or None

property height

Height of video frames.

Type int

property opened

Indicate whether the video is opened.

Type bool

property position

Current cursor position, indicating frame decoded.

Type int

read()

Read the next frame.

If the next frame have been decoded before and in the cache, then return it directly, otherwise decode, cache and return it.

返回 Return the frame if successful, otherwise None.

返回类型 ndarray or None

property resolution

Video resolution (width, height).

Type tuple

property vcap

The raw VideoCapture object.

Type cv2.VideoCapture

property width

Width of video frames.

Type int

19.1.2 Cache

class mmcv.video.**Cache**(*capacity*)

frames2video	Read the frame images from a directory and join them as a video.

19.1.3 mmcv.video.frames2video

mmcv.video.**frames2video**(*frame_dir: str*, *video_file: str*, *fps: float = 30*, *fourcc: str = 'XVID'*, *filename_tmpl: str = '{:06d}.jpg'*, *start: int = 0*, *end: int = 0*, *show_progress: bool = True*) → None

Read the frame images from a directory and join them as a video.

参数

- **frame_dir** (*str*) –The directory containing video frames.
- **video_file** (*str*) –Output filename.
- **fps** (*float*) –FPS of the output video.
- **fourcc** (*str*) –Fourcc of the output video, this should be compatible with the output file type.
- **filename_tmpl** (*str*) –Filename template with the index as the variable.
- **start** (*int*) –Starting frame index.
- **end** (*int*) –Ending frame index.
- **show_progress** (*bool*) –Whether to show a progress bar.

19.2 Optical Flow

dequantize_flow	Recover from quantized flow.
flow_from_bytes	Read dense optical flow from bytes.
flow_warp	Use flow to warp img.
flowread	Read an optical flow map.
flowwrite	Write optical flow to file.

下页继续

表 3 – 续上页

quantize_flow	Quantize flow to [0, 255].
sparse_flow_from_bytes	Read the optical flow in KITTI datasets from bytes.

19.2.1 mmcv.video.dequantize_flow

mmcv.video.**dequantize_flow**(*dx: numpy.ndarray*, *dy: numpy.ndarray*, *max_val: float = 0.02*, *denorm: bool = True*) → numpy.ndarray

Recover from quantized flow.

参数

- **dx** (*ndarray*) –Quantized dx.
- **dy** (*ndarray*) –Quantized dy.
- **max_val** (*float*) –Maximum value used when quantizing.
- **denorm** (*bool*) –Whether to multiply flow values with width/height.

返回 Dequantized flow.

返回类型 ndarray

19.2.2 mmcv.video.flow_from_bytes

mmcv.video.**flow_from_bytes**(*content: bytes*) → numpy.ndarray

Read dense optical flow from bytes.

注解: This load optical flow function works for FlyingChairs, FlyingThings3D, Sintel, FlyingChairsOcc datasets, but cannot load the data from ChairsSDHom.

参数 **content** (*bytes*) –Optical flow bytes got from files or other streams.

返回 Loaded optical flow with the shape (H, W, 2).

返回类型 ndarray

19.2.3 mmcv.video.flow_warp

mmcv.video.**flow_warp**(*img: numpy.ndarray*, *flow: numpy.ndarray*, *filling_value: int = 0*, *interpolate_mode: str = 'nearest'*) → numpy.ndarray

Use flow to warp img.

参数

- **img** (*ndarray*) –Image to be warped.
- **flow** (*ndarray*) –Optical Flow.
- **filling_value** (*int*) –The missing pixels will be set with filling_value.
- **interpolate_mode** (*str*) –bilinear -> Bilinear Interpolation; nearest -> Nearest Neighbor.

返回 Warped image with the same shape of img

返回类型 ndarray

19.2.4 mmcv.video.flowread

mmcv.video.**flowread**(*flow_or_path: Union[numpy.ndarray, str]*, *quantize: bool = False*, *concat_axis: int = 0*, **args*, ***kwargs*) → numpy.ndarray

Read an optical flow map.

参数

- **flow_or_path** (*ndarray or str*) –A flow map or filepath.
- **quantize** (*bool*) –whether to read quantized pair, if set to True, remaining args will be passed to *dequantize_flow()*.
- **concat_axis** (*int*) –The axis that dx and dy are concatenated, can be either 0 or 1. Ignored if quantize is False.

返回 Optical flow represented as a (h, w, 2) numpy array

返回类型 ndarray

19.2.5 mmcv.video.flowwrite

mmcv.video.**flowwrite**(*flow: numpy.ndarray*, *filename: str*, *quantize: bool = False*, *concat_axis: int = 0*, **args*, ***kwargs*) → None

Write optical flow to file.

If the flow is not quantized, it will be saved as a .flo file losslessly, otherwise a jpeg image which is lossy but of much smaller size. (dx and dy will be concatenated horizontally into a single image if quantize is True.)

参数

- **flow** (*ndarray*) –(h, w, 2) array of optical flow.
- **filename** (*str*) –Output filepath.
- **quantize** (*bool*) –Whether to quantize the flow and save it to 2 jpeg images. If set to True, remaining args will be passed to *quantize_flow()*.
- **concat_axis** (*int*) –The axis that dx and dy are concatenated, can be either 0 or 1. Ignored if quantize is False.

19.2.6 mmcv.video.quantize_flow

mmcv.video.**quantize_flow**(*flow: numpy.ndarray*, *max_val: float = 0.02*, *norm: bool = True*) → tuple

Quantize flow to [0, 255].

After this step, the size of flow will be much smaller, and can be dumped as jpeg images.

参数

- **flow** (*ndarray*) –(h, w, 2) array of optical flow.
- **max_val** (*float*) –Maximum value of flow, values beyond [-max_val, max_val] will be truncated.
- **norm** (*bool*) –Whether to divide flow values by image width/height.

返回 Quantized dx and dy.

返回类型 tuple[ndarray]

19.2.7 mmcv.video.sparse_flow_from_bytes

mmcv.video.**sparse_flow_from_bytes**(*content: bytes*) → Tuple[numpy.ndarray, numpy.ndarray]

Read the optical flow in KITTI datasets from bytes.

This function is modified from RAFT load the KITTI datasets.

参数 **content** (*bytes*) –Optical flow bytes got from files or other streams.

返回 Loaded optical flow with the shape (H, W, 2) and flow valid mask with the shape (H, W).

返回类型 Tuple(ndarray, ndarray)

19.3 Video Processing

concat_video	Concatenate multiple videos into a single one.
convert_video	Convert a video with ffmpeg.
cut_video	Cut a clip from a video.
resize_video	Resize a video.

19.3.1 mmcv.video.concat_video

mmcv.video.**concat_video**(*video_list: List*, *out_file: str*, *vcodec: Optional[str] = None*, *acodec: Optional[str] = None*, *log_level: str = 'info'*, *print_cmd: bool = False*) → None

Concatenate multiple videos into a single one.

参数

- **video_list** (*list*) –A list of video filenames
- **out_file** (*str*) –Output video filename
- **vcodec** (*None or str*) –Output video codec, None for unchanged
- **acodec** (*None or str*) –Output audio codec, None for unchanged
- **log_level** (*str*) –Logging level of ffmpeg.
- **print_cmd** (*bool*) –Whether to print the final ffmpeg command.

19.3.2 mmcv.video.convert_video

mmcv.video.**convert_video**(*in_file: str*, *out_file: str*, *print_cmd: bool = False*, *pre_options: str = ''*, ***kwargs*) → None

Convert a video with ffmpeg.

This provides a general api to ffmpeg, the executed command is:

```
`ffmpeg -y <pre_options> -i <in_file> <options> <out_file>`
```

Options(kwargs) are mapped to ffmpeg commands with the following rules:

- key=val: "-key val"
- key=True: "-key"
- key=False: ""

参数

- **in_file** (*str*) –Input video filename.
- **out_file** (*str*) –Output video filename.
- **pre_options** (*str*) –Options appears before “-i <in_file>”.
- **print_cmd** (*bool*) –Whether to print the final ffmpeg command.

19.3.3 mmcv.video.cut_video

mmcv.video.**cut_video**(*in_file: str*, *out_file: str*, *start: Optional[float] = None*, *end: Optional[float] = None*, *vcodec: Optional[str] = None*, *acodec: Optional[str] = None*, *log_level: str = 'info'*, *print_cmd: bool = False*) → None

Cut a clip from a video.

参数

- **in_file** (*str*) –Input video filename.
- **out_file** (*str*) –Output video filename.
- **start** (*None or float*) –Start time (in seconds).
- **end** (*None or float*) –End time (in seconds).
- **vcodec** (*None or str*) –Output video codec, None for unchanged.
- **acodec** (*None or str*) –Output audio codec, None for unchanged.
- **log_level** (*str*) –Logging level of ffmpeg.
- **print_cmd** (*bool*) –Whether to print the final ffmpeg command.

19.3.4 mmcv.video.resize_video

mmcv.video.**resize_video**(*in_file: str*, *out_file: str*, *size: Optional[tuple] = None*, *ratio: Optional[Union[tuple, float]] = None*, *keep_ar: bool = False*, *log_level: str = 'info'*, *print_cmd: bool = False*) → None

Resize a video.

参数

- **in_file** (*str*) –Input video filename.
- **out_file** (*str*) –Output video filename.
- **size** (*tuple*) –Expected size (w, h), eg, (320, 240) or (320, -1).
- **ratio** (*tuple or float*) –Expected resize ratio, (2, 0.5) means (w*2, h*0.5).
- **keep_ar** (*bool*) –Whether to keep original aspect ratio.
- **log_level** (*str*) –Logging level of ffmpeg.

- **print_cmd** (*bool*) –Whether to print the final ffmpeg command.

CHAPTER 20

mmcv.visualization

mmcv.visualization

20.1 Color

Color	An enum that defines common colors.

20.1.1 Color

class mmcv.visualization.**Color**(*value*)

An enum that defines common colors.

Contains red, green, blue, cyan, yellow, magenta, white and black.

color_val	Convert various input to color tuples.

20.1.2 mmcv.visualization.color_val

mmcv.visualization.**color_val**(*color: Union[*mmcv.visualization.color.Color*, str, tuple, int, numpy.ndarray]*) → tuple

Convert various input to color tuples.

参数 **color** (*Color*/str/tuple/int/ndarray) –Color inputs

返回 A tuple of 3 integers indicating BGR channels.

返回类型 tuple[int]

20.2 Image

imshow	Show an image.
imshow_bboxes	Draw bboxes on an image.
imshow_det_bboxes	Draw bboxes and class labels (with scores) on an image.

20.2.1 mmcv.visualization.imshow

mmcv.visualization.**imshow**(*img: Union[str, numpy.ndarray]*, *win_name: str = ''*, *wait_time: int = 0*)

Show an image.

参数

- **img** (*str or ndarray*) –The image to be displayed.
- **win_name** (*str*) –The window name.
- **wait_time** (*int*) –Value of waitKey param.

20.2.2 mmcv.visualization.imshow_bboxes

mmcv.visualization.**imshow_bboxes**(*img: Union[str, numpy.ndarray]*, *bboxes: Union[list, numpy.ndarray]*, *colors: Union[*mmcv.visualization.color.Color*, str, tuple, int, numpy.ndarray] = 'green'*, *top_k: int = -1*, *thickness: int = 1*, *show: bool = True*, *win_name: str = ''*, *wait_time: int = 0*, *out_file: Optional[str] = None*)

Draw bboxes on an image.

参数

- **img** (*str or ndarray*) –The image to be displayed.
- **bboxes** (*list or ndarray*) –A list of ndarray of shape (k, 4).

- **colors** (*Color or str or tuple or int or ndarray*) –A list of colors.
- **top_k** (*int*) –Plot the first k bboxes only if set positive.
- **thickness** (*int*) –Thickness of lines.
- **show** (*bool*) –Whether to show the image.
- **win_name** (*str*) –The window name.
- **wait_time** (*int*) –Value of waitKey param.
- **out_file** (*str, optional*) –The filename to write the image.

返回 The image with bboxes drawn on it.

返回类型 ndarray

20.2.3 mmcv.visualization.imshow_det_bboxes

mmcv.visualization.**imshow_det_bboxes**(*img: Union[str, numpy.ndarray]*, *bboxes: numpy.ndarray*, *labels: numpy.ndarray*, *class_names: Optional[List[str]] = None*, *score_thr: float = 0*, *bbox_color: Union[*mmcv.visualization.color.Color*, str, tuple, int, numpy.ndarray] = 'green'*, *text_color: Union[*mmcv.visualization.color.Color*, str, tuple, int, numpy.ndarray] = 'green'*, *thickness: int = 1*, *font_scale: float = 0.5*, *show: bool = True*, *win_name: str = ''*, *wait_time: int = 0*, *out_file: Optional[str] = None*)

Draw bboxes and class labels (with scores) on an image.

参数

- **img** (*str or ndarray*) –The image to be displayed.
- **bboxes** (*ndarray*) –Bounding boxes (with scores), shaped (n, 4) or (n, 5).
- **labels** (*ndarray*) –Labels of bboxes.
- **class_names** (*list[str]*) –Names of each classes.
- **score_thr** (*float*) –Minimum score of bboxes to be shown.
- **bbox_color** (*Color or str or tuple or int or ndarray*) –Color of bbox lines.
- **text_color** (*Color or str or tuple or int or ndarray*) –Color of texts.
- **thickness** (*int*) –Thickness of lines.
- **font_scale** (*float*) –Font scales of texts.
- **show** (*bool*) –Whether to show the image.

- **win_name** (*str*) –The window name.
- **wait_time** (*int*) –Value of waitKey param.
- **out_file** (*str or None*) –The filename to write the image.

返回 The image with bboxes drawn on it.

返回类型 ndarray

20.3 Optical Flow

flow2rgb	Convert flow map to RGB image.
flowshow	Show optical flow.
make_color_wheel	Build a color wheel.

20.3.1 mmcv.visualization.flow2rgb

mmcv.visualization.**flow2rgb**(*flow: numpy.ndarray*, *color_wheel: Optional[numpy.ndarray] = None*, *unknown_thr: float = 1000000.0*) → numpy.ndarray

Convert flow map to RGB image.

参数

- **flow** (*ndarray*) –Array of optical flow.
- **color_wheel** (*ndarray or None*) –Color wheel used to map flow field to RGB colorspace. Default color wheel will be used if not specified.
- **unknown_thr** (*float*) –Values above this threshold will be marked as unknown and thus ignored.

返回 RGB image that can be visualized.

返回类型 ndarray

20.3.2 mmcv.visualization.flowshow

mmcv.visualization.**flowshow**(*flow: Union[numpy.ndarray, str]*, *win_name: str = ''*, *wait_time: int = 0*) → None

Show optical flow.

参数

- **flow** (*ndarray or str*) –The optical flow to be displayed.
- **win_name** (*str*) –The window name.

- **wait_time** (*int*) –Value of waitKey param.

20.3.3 mmcv.visualization.make_color_wheel

mmcv.visualization.**make_color_wheel**(*bins: Optional[Union[list, tuple]] = None*) → numpy.ndarray

Build a color wheel.

参数 **bins** (*list or tuple, optional*) –Specify the number of bins for each color range, corresponding to six ranges: red -> yellow, yellow -> green, green -> cyan, cyan -> blue, blue -> magenta, magenta -> red. [15, 6, 4, 11, 13, 6] is used for default (see Middlebury).

返回 Color wheel of shape (total_bins, 3).

返回类型 ndarray

CHAPTER 21

mmcv.cnn

mmcv.cnn

21.1 Module

`ContextBlock`	ContextBlock module in GCNet.
`Conv2d`	
`Conv3d`	
`ConvAWS2d`	AWS (Adaptive Weight Standardization)
`ConvModule`	A conv block that bundles conv/norm/activation layers.
`ConvTranspose2d`	

下页继续

表 1 – 续上页

ConvTranspose3d	
ConvWS2d	
DepthwiseSeparableConvModule	Depthwise separable convolution module.
GeneralizedAttention	GeneralizedAttention module.
HSigmoid	Hard Sigmoid Module.
HSwish	Hard Swish Module.
Linear	
MaxPool2d	
MaxPool3d	
NonLocal1d	1D Non-local module.
NonLocal2d	2D Non-local module.
NonLocal3d	3D Non-local module.
Scale	A learnable scale parameter.
Swish	Swish Module.
Conv2dRFSearchOp	Enable Conv2d with receptive field searching ability.

21.1.1 ContextBlock

class mmcv.cnn.**ContextBlock**(*in_channels: int*, *ratio: float*, *pooling_type: str = 'att'*, *fusion_types: tuple = ('channel_add')*)

ContextBlock module in GCNet.

See ‘GCNet: Non-local Networks Meet Squeeze-Excitation Networks and Beyond’ (https://arxiv.org/abs/1904.11492) for details.

参数

- **in_channels** (*int*) –Channels of the input feature map.
- **ratio** (*float*) –Ratio of channels of transform bottleneck
- **pooling_type** (*str*) –Pooling method for context modeling. Options are ‘att’ and ‘avg’, stand for attention pooling and average pooling respectively. Default: ‘att’.
- **fusion_types** (*Sequence[str]*) –Fusion method for feature fusion, Options are ‘channels_add’, ‘channel_mul’, stand for channelwise addition and multiplication respectively. Default: (‘channel_add’,)

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.2 Conv2d

class mmcv.cnn.**Conv2d**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, int]]*, *stride: Union[int, Tuple[int, int]] = 1*, *padding: Union[str, int, Tuple[int, int]] = 0*, *dilation: Union[int, Tuple[int, int]] = 1*, *groups: int = 1*, *bias: bool = True*, *padding_mode: str = 'zeros'*, *device=None*, *dtype=None*)

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.3 Conv3d

class mmcv.cnn.**Conv3d**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, int, int]]*, *stride: Union[int, Tuple[int, int, int]] = 1*, *padding: Union[str, int, Tuple[int, int, int]] = 0*, *dilation: Union[int, Tuple[int, int, int]] = 1*, *groups: int = 1*, *bias: bool = True*, *padding_mode: str = 'zeros'*, *device=None*, *dtype=None*)

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.4 ConvAWS2d

class mmcv.cnn.**ConvAWS2d**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, int]]*, *stride: Union[int, Tuple[int, int]] = 1*, *padding: Union[int, Tuple[int, int]] = 0*, *dilation: Union[int, Tuple[int, int]] = 1*, *groups: int = 1*, *bias: bool = True*)

AWS (Adaptive Weight Standardization)

This is a variant of Weight Standardization (https://arxiv.org/pdf/1903.10520.pdf) It is used in DetectoRS to avoid NaN (https://arxiv.org/pdf/2006.02334.pdf)

参数

- **in_channels** (*int*) –Number of channels in the input image
- **out_channels** (*int*) –Number of channels produced by the convolution
- **kernel_size** (*int or tuple*) –Size of the conv kernel
- **stride** (*int or tuple, optional*) –Stride of the convolution. Default: 1
- **padding** (*int or tuple, optional*) –Zero-padding added to both sides of the input. Default: 0
- **dilation** (*int or tuple, optional*) –Spacing between kernel elements. Default: 1
- **groups** (*int, optional*) –Number of blocked connections from input channels to output channels. Default: 1
- **bias** (*bool, optional*) –If set True, adds a learnable bias to the output. Default: True

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.5 ConvModule

class mmcv.cnn.**ConvModule**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, int]]*, *stride: Union[int, Tuple[int, int]] = 1*, *padding: Union[int, Tuple[int, int]] = 0*, *dilation: Union[int, Tuple[int, int]] = 1*, *groups: int = 1*, *bias: Union[bool, str] = 'auto'*, *conv_cfg: Optional[Dict] = None*, *norm_cfg: Optional[Dict] = None*, *act_cfg: Optional[Dict] = {'type': 'ReLU'}*, *inplace: bool = True*, *with_spectral_norm: bool = False*, *padding_mode: str = 'zeros'*, *order: tuple = ('conv', 'norm', 'act')*)

A conv block that bundles conv/norm/activation layers.

This block simplifies the usage of convolution layers, which are commonly used with a norm layer (e.g., BatchNorm) and activation layer (e.g., ReLU). It is based upon three build methods: *build_conv_layer()*, *build_norm_layer()* and *build_activation_layer()*.

Besides, we add some additional features in this module. 1. Automatically set *bias* of the conv layer. 2. Spectral norm is supported. 3. More padding modes are supported. Before PyTorch 1.5, nn.Conv2d only supports zero and circular padding, and we add "reflect" padding mode.

参数

- **in_channels** (*int*) –Number of channels in the input feature map. Same as that in `nn._ConvNd`.
- **out_channels** (*int*) –Number of channels produced by the convolution. Same as that in `nn._ConvNd`.
- **kernel_size** (*int | tuple[int]*) –Size of the convolving kernel. Same as that in `nn._ConvNd`.
- **stride** (*int | tuple[int]*) –Stride of the convolution. Same as that in `nn._ConvNd`.
- **padding** (*int | tuple[int]*) –Zero-padding added to both sides of the input. Same as that in `nn._ConvNd`.
- **dilation** (*int | tuple[int]*) –Spacing between kernel elements. Same as that in `nn._ConvNd`.
- **groups** (*int*) –Number of blocked connections from input channels to output channels. Same as that in `nn._ConvNd`.
- **bias** (*bool | str*) –If specified as *auto*, it will be decided by the norm_cfg. Bias will be set as True if *norm_cfg* is None, otherwise False. Default: "auto".
- **conv_cfg** (*dict*) –Config dict for convolution layer. Default: None, which means using conv2d.
- **norm_cfg** (*dict*) –Config dict for normalization layer. Default: None.

- **act_cfg** (*dict*) –Config dict for activation layer. Default: dict(type='ReLU').
- **inplace** (*bool*) –Whether to use inplace mode for activation. Default: True.
- **with_spectral_norm** (*bool*) –Whether use spectral norm in conv module. Default: False.
- **padding_mode** (*str*) –If the *padding_mode* has not been supported by current *Conv2d* in PyTorch, we will use our own padding layer instead. Currently, we support ['zeros', 'circular'] with official implementation and ['reflect'] with our own implementation. Default: 'zeros'.
- **order** (*tuple[str]*) –The order of conv/norm/activation layers. It is a sequence of "conv", "norm" and "act". Common examples are ("conv", "norm", "act") and ("act", "conv", "norm"). Default: ('conv', 'norm', 'act').

forward(*x: torch.Tensor*, *activate: bool = True*, *norm: bool = True*) → torch.Tensor

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.

21.1.6 ConvTranspose2d

class mmcv.cnn.**ConvTranspose2d**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, int]]*, *stride: Union[int, Tuple[int, int]] = 1*, *padding: Union[int, Tuple[int, int]] = 0*, *output_padding: Union[int, Tuple[int, int]] = 0*, *groups: int = 1*, *bias: bool = True*, *dilation: Union[int, Tuple[int, int]] = 1*, *padding_mode: str = 'zeros'*, *device=None*, *dtype=None*)

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.7 ConvTranspose3d

class mmcv.cnn.**ConvTranspose3d**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, int, int]]*, *stride: Union[int, Tuple[int, int, int]] = 1*, *padding: Union[int, Tuple[int, int, int]] = 0*, *output_padding: Union[int, Tuple[int, int, int]] = 0*, *groups: int = 1*, *bias: bool = True*, *dilation: Union[int, Tuple[int, int, int]] = 1*, *padding_mode: str = 'zeros'*, *device=None*, *dtype=None*)

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.8 ConvWS2d

class mmcv.cnn.**ConvWS2d**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, int]]*, *stride: Union[int, Tuple[int, int]] = 1*, *padding: Union[int, Tuple[int, int]] = 0*, *dilation: Union[int, Tuple[int, int]] = 1*, *groups: int = 1*, *bias: bool = True*, *eps: float = 1e-05*)

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.9 DepthwiseSeparableConvModule

class mmcv.cnn.**DepthwiseSeparableConvModule**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, int]]*, *stride: Union[int, Tuple[int, int]] = 1*, *padding: Union[int, Tuple[int, int]] = 0*, *dilation: Union[int, Tuple[int, int]] = 1*, *norm_cfg: Optional[Dict] = None*, *act_cfg: Dict = {'type': 'ReLU'}*, *dw_norm_cfg: Union[Dict, str] = 'default'*, *dw_act_cfg: Union[Dict, str] = 'default'*, *pw_norm_cfg: Union[Dict, str] = 'default'*, *pw_act_cfg: Union[Dict, str] = 'default'*, ***kwargs*)

Depthwise separable convolution module.

See https://arxiv.org/pdf/1704.04861.pdf for details.

This module can replace a ConvModule with the conv block replaced by two conv block: depthwise conv block and pointwise conv block. The depthwise conv block contains depthwise-conv/norm/activation layers. The pointwise conv block contains pointwise-conv/norm/activation layers. It should be noted that there will be norm/activation layer in the depthwise conv block if *norm_cfg* and *act_cfg* are specified.

参数

- **in_channels** (*int*) –Number of channels in the input feature map. Same as that in `nn._ConvNd`.
- **out_channels** (*int*) –Number of channels produced by the convolution. Same as that in `nn._ConvNd`.
- **kernel_size** (*int | tuple[int]*) –Size of the convolving kernel. Same as that in `nn._ConvNd`.
- **stride** (*int | tuple[int]*) –Stride of the convolution. Same as that in `nn._ConvNd`. Default: 1.
- **padding** (*int | tuple[int]*) –Zero-padding added to both sides of the input. Same as that in `nn._ConvNd`. Default: 0.
- **dilation** (*int | tuple[int]*) –Spacing between kernel elements. Same as that in `nn._ConvNd`. Default: 1.
- **norm_cfg** (*dict*) –Default norm config for both depthwise ConvModule and pointwise ConvModule. Default: None.
- **act_cfg** (*dict*) –Default activation config for both depthwise ConvModule and pointwise ConvModule. Default: dict(type='ReLU').
- **dw_norm_cfg** (*dict*) –Norm config of depthwise ConvModule. If it is 'default', it will be the same as *norm_cfg*. Default: 'default'.

- **dw_act_cfg** (*dict*) –Activation config of depthwise ConvModule. If it is ‘default’, it will be the same as *act_cfg*. Default: ‘default’.
- **pw_norm_cfg** (*dict*) –Norm config of pointwise ConvModule. If it is ‘default’, it will be the same as *norm_cfg*. Default: ‘default’.
- **pw_act_cfg** (*dict*) –Activation config of pointwise ConvModule. If it is ‘default’, it will be the same as *act_cfg*. Default: ‘default’.
- **kwargs** (*optional*) –Other shared arguments for depthwise and pointwise ConvModule. See ConvModule for ref.

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.10 GeneralizedAttention

class mmcv.cnn.**GeneralizedAttention**(*in_channels: int*, *spatial_range: int = - 1*, *num_heads: int = 9*, *position_embedding_dim: int = - 1*, *position_magnitude: int = 1*, *kv_stride: int = 2*, *q_stride: int = 1*, *attention_type: str = '1111'*)

GeneralizedAttention module.

See ‘An Empirical Study of Spatial Attention Mechanisms in Deep Networks’ (https://arxiv.org/abs/1904.05873) for details.

参数

- **in_channels** (*int*) –Channels of the input feature map.
- **spatial_range** (*int*) –The spatial range. -1 indicates no spatial range constraint. Default: -1.
- **num_heads** (*int*) –The head number of empirical_attention module. Default: 9.
- **position_embedding_dim** (*int*) –The position embedding dimension. Default: -1.
- **position_magnitude** (*int*) –A multiplier acting on coord difference. Default: 1.
- **kv_stride** (*int*) –The feature stride acting on key/value feature map. Default: 2.
- **q_stride** (*int*) –The feature stride acting on query feature map. Default: 1.

- **attention_type** (*str*) –A binary indicator string for indicating which items in generalized empirical_attention module are used. Default: '1111'.
  - '1000' indicates 'query and key content' (appr - appr) item,
  - '0100' indicates 'query content and relative position' (appr - position) item,
  - '0010' indicates 'key content only' (bias - appr) item,
  - '0001' indicates 'relative position only' (bias - position) item.

forward(*x_input: torch.Tensor*) → torch.Tensor

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.

21.1.11 HSigmoid

class mmcv.cnn.**HSigmoid**(*bias: float = 3.0*, *divisor: float = 6.0*, *min_value: float = 0.0*, *max_value: float = 1.0*)

Hard Sigmoid Module. Apply the hard sigmoid function: Hsigmoid(x) = min(max((x + bias) / divisor, min_value), max_value) Default: Hsigmoid(x) = min(max((x + 3) / 6, 0), 1)

注解: In MMCV v1.4.4, we modified the default value of args to align with PyTorch official.

参数

- **bias** (*float*) –Bias of the input feature map. Default: 3.0.
- **divisor** (*float*) –Divisor of the input feature map. Default: 6.0.
- **min_value** (*float*) –Lower bound value. Default: 0.0.
- **max_value** (*float*) –Upper bound value. Default: 1.0.

返回 The output tensor.

返回类型 Tensor

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.12 HSwish

class mmcv.cnn.**HSwish**(*inplace: bool = False*)

Hard Swish Module.

This module applies the hard swish function:

$$Hswish(x) = x * ReLU6(x + 3)/6$$

参数 **inplace** (*bool*) –can optionally do the operation in-place. Default: False.

返回 The output tensor.

返回类型 Tensor

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.13 Linear

class mmcv.cnn.**Linear**(*in_features: int*, *out_features: int*, *bias: bool = True*, *device=None*, *dtype=None*)

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.14 MaxPool2d

class mmcv.cnn.**MaxPool2d**(*kernel_size: Union[int, Tuple[int, ...]]*, *stride: Optional[Union[int, Tuple[int, ...]]] = None*, *padding: Union[int, Tuple[int, ...]] = 0*, *dilation: Union[int, Tuple[int, ...]] = 1*, *return_indices: bool = False*, *ceil_mode: bool = False*)

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.15 MaxPool3d

class mmcv.cnn.**MaxPool3d**(*kernel_size: Union[int, Tuple[int, ...]]*, *stride: Optional[Union[int, Tuple[int, ...]]] = None*, *padding: Union[int, Tuple[int, ...]] = 0*, *dilation: Union[int, Tuple[int, ...]] = 1*, *return_indices: bool = False*, *ceil_mode: bool = False*)

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.16 NonLocal1d

class mmcv.cnn.**NonLocal1d**(*in_channels: int*, *sub_sample: bool = False*, *conv_cfg: Dict = {'type': 'Conv1d'}*, ***kwargs*)

1D Non-local module.

参数

- **in_channels** (*int*) –Same as *NonLocalND*.

- **sub_sample** (*bool*) –Whether to apply max pooling after pairwise function (Note that the *sub_sample* is applied on spatial only). Default: False.
- **conv_cfg** (*None | dict*) –Same as *NonLocalND*. Default: dict(type=’Conv1d’).

21.1.17 NonLocal2d

class mmcv.cnn.**NonLocal2d**(*in_channels: int*, *sub_sample: bool = False*, *conv_cfg: Dict = {'type': 'Conv2d'}*, ***kwargs*)

2D Non-local module.

参数

- **in_channels** (*int*) –Same as *NonLocalND*.
- **sub_sample** (*bool*) –Whether to apply max pooling after pairwise function (Note that the *sub_sample* is applied on spatial only). Default: False.
- **conv_cfg** (*None | dict*) –Same as *NonLocalND*. Default: dict(type=’Conv2d’).

21.1.18 NonLocal3d

class mmcv.cnn.**NonLocal3d**(*in_channels: int*, *sub_sample: bool = False*, *conv_cfg: Dict = {'type': 'Conv3d'}*, ***kwargs*)

3D Non-local module.

参数

- **in_channels** (*int*) –Same as *NonLocalND*.
- **sub_sample** (*bool*) –Whether to apply max pooling after pairwise function (Note that the *sub_sample* is applied on spatial only). Default: False.
- **conv_cfg** (*None | dict*) –Same as *NonLocalND*. Default: dict(type=’Conv3d’).

21.1.19 Scale

class mmcv.cnn.**Scale**(*scale: float = 1.0*)

A learnable scale parameter.

This layer scales the input by a learnable factor. It multiplies a learnable scale parameter of shape (1,) with input of any shape.

参数 **scale** (*float*) –Initial value of scale factor. Default: 1.0

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.20 Swish

class mmcv.cnn.**Swish**

Swish Module.

This module applies the swish function:

$$Swish(x) = x * Sigmoid(x)$$

返回 The output tensor.

返回类型 Tensor

forward(*x: torch.Tensor*) → torch.Tensor

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.

21.1.21 Conv2dRFSearchOp

class mmcv.cnn.**Conv2dRFSearchOp**(*op_layer: torch.nn.modules.module.Module*, *global_config: dict*, *verbose: bool = True*)

Enable Conv2d with receptive field searching ability.

参数

- **op_layer** (*nn.Module*) –pytorch module, e,g, Conv2d
- **global_config** (*dict*) –config dict. Defaults to None. By default this must include:
  - ” init_alphas” : The value for initializing weights of each branch.
  - ” num_branches” : The controller of the size of search space (the number of branches).
  - ” exp_rate” : The controller of the sparsity of search space.
  - ” mmin” : The minimum dilation rate.

– ” mmax” : The maximum dilation rate.

Extra keys may exist, but are used by RFSearchHook, e.g., “step” , “max_step” , “search_interval” , and “skip_layer” .

- **verbose** (*bool*) –Determines whether to print rf-next related logging messages. Defaults to True.

estimate_rates() → None

Estimate new dilation rate based on trained branch_weights.

expand_rates() → None

Expand dilation rate.

forward(*input: torch.Tensor*) → torch.Tensor

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.

21.2 Build Function

build_activation_layer	Build activation layer.
build_conv_layer	Build convolution layer.
build_norm_layer	Build normalization layer.
build_padding_layer	Build padding layer.
build_plugin_layer	Build plugin layer.
build_upsample_layer	Build upsample layer.

21.2.1 mmcv.cnn.build_activation_layer

mmcv.cnn.**build_activation_layer**(*cfg: Dict*) → torch.nn.modules.module.Module

Build activation layer.

参数 **cfg** (*dict*) –The activation layer config, which should contain:

- type (str): Layer type.
- layer args: Args needed to instantiate an activation layer.

返回 Created activation layer.

返回类型 nn.Module

21.2.2 mmcv.cnn.build_conv_layer

mmcv.cnn.**build_conv_layer**(*cfg: Optional[Dict]*, **args*, ***kwargs*) → torch.nn.modules.module.Module

Build convolution layer.

参数

- **cfg** (*None or dict*) –The conv layer config, which should contain: - type (str): Layer type. - layer args: Args needed to instantiate an conv layer.
- **args** (*argument list*) –Arguments passed to the __init__ method of the corresponding conv layer.
- **kwargs** (*keyword arguments*) –Keyword arguments passed to the __init__ method of the corresponding conv layer.

返回 Created conv layer.

返回类型 nn.Module

21.2.3 mmcv.cnn.build_norm_layer

mmcv.cnn.**build_norm_layer**(*cfg: Dict*, *num_features: int*, *postfix: Union[int, str] = ''*) → Tuple[str, torch.nn.modules.module.Module]

Build normalization layer.

参数

- **cfg** (*dict*) –The norm layer config, which should contain:
  - type (str): Layer type.
  - layer args: Args needed to instantiate a norm layer.
  - requires_grad (bool, optional): Whether stop gradient updates.
- **num_features** (*int*) –Number of input channels.
- **postfix** (*int | str*) –The postfix to be appended into norm abbreviation to create named layer.

返回 The first element is the layer name consisting of abbreviation and postfix, e.g., bn1, gn. The second element is the created norm layer.

返回类型 tuple[str, nn.Module]

21.2.4 mmcv.cnn.build_padding_layer

mmcv.cnn.**build_padding_layer**(*cfg: Dict*, **args*, ***kwargs*) → torch.nn.modules.module.Module

Build padding layer.

参数 **cfg** (*dict*) –The padding layer config, which should contain: - type (str): Layer type. - layer args: Args needed to instantiate a padding layer.

返回 Created padding layer.

返回类型 nn.Module

21.2.5 mmcv.cnn.build_plugin_layer

mmcv.cnn.**build_plugin_layer**(*cfg: Dict*, *postfix: Union[int, str] = ''*, ***kwargs*) → Tuple[str, torch.nn.modules.module.Module]

Build plugin layer.

参数

- **cfg** (*dict*) –cfg should contain:
  - type (str): identify plugin layer type.
  - layer args: args needed to instantiate a plugin layer.
- **postfix** (*int, str*) –appended into norm abbreviation to create named layer. Default: ''.

返回 The first one is the concatenation of abbreviation and postfix. The second is the created plugin layer.

返回类型 tuple[str, nn.Module]

21.2.6 mmcv.cnn.build_upsample_layer

mmcv.cnn.**build_upsample_layer**(*cfg: Dict*, **args*, ***kwargs*) → torch.nn.modules.module.Module

Build upsample layer.

参数

- **cfg** (*dict*) –The upsample layer config, which should contain:
  - type (str): Layer type.
  - scale_factor (int): Upsample ratio, which is not applicable to deconv.
  - layer args: Args needed to instantiate a upsample layer.
- **args** (*argument list*) –Arguments passed to the `__init__` method of the corresponding conv layer.

- **kwargs** (*keyword arguments*) –Keyword arguments passed to the __init__ method of the corresponding conv layer.

返回 Created upsample layer.

返回类型 nn.Module

21.3 Miscellaneous

fuse_conv_bn	Recursively fuse conv and bn in a module.
conv_ws_2d	
is_norm	Check if a layer is a normalization layer.
make_res_layer	
make_vgg_layer	
get_model_complexity_info	Get complexity information of a model.

21.3.1 mmcv.cnn.fuse_conv_bn

mmcv.cnn.**fuse_conv_bn**(*module: torch.nn.modules.module.Module*) → torch.nn.modules.module.Module

Recursively fuse conv and bn in a module.

During inference, the functionary of batch norm layers is turned off but only the mean and var alone channels are used, which exposes the chance to fuse it with the preceding conv layers to save computations and simplify network structures.

参数 **module** (*nn.Module*) –Module to be fused.

返回 Fused module.

返回类型 nn.Module

21.3.2 mmcv.cnn.conv_ws_2d

mmcv.cnn.**conv_ws_2d**(*input: torch.Tensor*, *weight: torch.Tensor*, *bias: Optional[torch.Tensor] = None*, *stride: Union[int, Tuple[int, int]] = 1*, *padding: Union[int, Tuple[int, int]] = 0*, *dilation: Union[int, Tuple[int, int]] = 1*, *groups: int = 1*, *eps: float = 1e-05*) → torch.Tensor

21.3.3 mmcv.cnn.is_norm

mmcv.cnn.**is_norm**(*layer: torch.nn.modules.module.Module*, *exclude: Optional[Union[type, tuple]] = None*) → bool

Check if a layer is a normalization layer.

参数

- **layer** (*nn.Module*) –The layer to be checked.
- **exclude** (*type | tuple[type]*) –Types to be excluded.

返回 Whether the layer is a norm layer.

返回类型 bool

21.3.4 mmcv.cnn.make_res_layer

mmcv.cnn.**make_res_layer**(*block: torch.nn.modules.module.Module*, *inplanes: int*, *planes: int*, *blocks: int*, *stride: int = 1*, *dilation: int = 1*, *style: str = 'pytorch'*, *with_cp: bool = False*) → torch.nn.modules.module.Module

21.3.5 mmcv.cnn.make_vgg_layer

mmcv.cnn.**make_vgg_layer**(*inplanes: int*, *planes: int*, *num_blocks: int*, *dilation: int = 1*, *with_bn: bool = False*, *ceil_mode: bool = False*) → List[torch.nn.modules.module.Module]

21.3.6 mmcv.cnn.get_model_complexity_info

mmcv.cnn.**get_model_complexity_info**(*model: torch.nn.modules.module.Module*, *input_shape: tuple*, *print_per_layer_stat: bool = True*, *as_strings: bool = True*, *input_constructor: Optional[Callable] = None*, *flush: bool = False*, *ost: TextIO = <_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*) → tuple

Get complexity information of a model.

This method can calculate FLOPs and parameter counts of a model with corresponding input shape. It can also print complexity information for each layer in a model.

Supported layers are listed as below:

- Convolutions: `nn.Conv1d`, `nn.Conv2d`, `nn.Conv3d`.
- Activations: `nn.ReLU`, `nn.PReLU`, `nn.ELU`, `nn.LeakyReLU`, `nn.ReLU6`.
- Poolings: `nn.MaxPool1d`, `nn.MaxPool2d`, `nn.MaxPool3d`, `nn.AvgPool1d`, `nn.AvgPool2d`, `nn.AvgPool3d`, `nn.AdaptiveMaxPool1d`, `nn.AdaptiveMaxPool2d`,

nn.AdaptiveMaxPool3d, nn.AdaptiveAvgPool1d, nn.AdaptiveAvgPool2d, nn.AdaptiveAvgPool3d.

- BatchNorms: nn.BatchNorm1d, nn.BatchNorm2d, nn.BatchNorm3d, nn.GroupNorm, nn.InstanceNorm1d, InstanceNorm2d, InstanceNorm3d, nn.LayerNorm.
- Linear: nn.Linear.
- Deconvolution: nn.ConvTranspose2d.
- Upsample: nn.Upsample.

参数

- **model** (*nn.Module*) –The model for complexity calculation.
- **input_shape** (*tuple*) –Input shape used for calculation.
- **print_per_layer_stat** (*bool*) –Whether to print complexity information for each layer in a model. Default: True.
- **as_strings** (*bool*) –Output FLOPs and params counts in a string form. Default: True.
- **input_constructor** (*None | callable*) –If specified, it takes a callable method that generates input. otherwise, it will generate a random tensor with input shape to calculate FLOPs. Default: None.
- **flush** (*bool*) –same as that in print(). Default: False.
- **ost** (*stream*) –same as file param in print(). Default: sys.stdout.

返回 If as_strings is set to True, it will return FLOPs and parameter counts in a string format. otherwise, it will return those in a float number format.

返回类型 tuple[float | str]

CHAPTER 22

mmcv.ops

BorderAlign	Border align pooling layer.
CARAFE	CARAFE: Content-Aware ReAssembly of FEatures
CARAFENaive	
CARAFEPack	A unified package of CARAFE upsampler that contains: 1) channel compressor 2) content encoder 3) CARAFE op.
Conv2d	alias of `mmcv.ops.deprecated_wrappers.Conv2d_deprecated`
ConvTranspose2d	alias of `mmcv.ops.deprecated_wrappers.ConvTranspose2d_deprecated`
CornerPool	Corner Pooling.
Correlation	Correlation operator
CrissCrossAttention	Criss-Cross Attention Module.
DeformConv2d	Deformable 2D convolution.
DeformConv2dPack	A Deformable Conv Encapsulation that acts as normal Conv layers.
DeformRoIPool	
DeformRoIPoolPack	

下页继续

表 1 – 续上页

DynamicScatter	Scatters points into voxels, used in the voxel encoder with dynamic voxelization.
FusedBiasLeakyReLU	Fused bias leaky ReLU.
GroupAll	Group xyz with feature.
Linear	alias of `mmcv.ops.deprecated_wrappers.Linear_deprecated`
MaskedConv2d	A MaskedConv2d which inherits the official Conv2d.
MaxPool2d	alias of `mmcv.ops.deprecated_wrappers.MaxPool2d_deprecated`
ModulatedDeformConv2d	
ModulatedDeformConv2dPack	A ModulatedDeformable Conv Encapsulation that acts as normal Conv layers.
ModulatedDeformRoIPoolPack	
MultiScaleDeformableAttention	An attention module used in Deformable-Detr.
PSAMask	
PointsSampler	Points sampling.
PrRoIPool	The operation of precision RoI pooling.
QueryAndGroup	Groups points with a ball query of radius.
RiRoIAlignRotated	Rotation-invariant RoI align pooling layer for rotated proposals.
RoIAlign	RoI align pooling layer.
RoIAlignRotated	RoI align pooling layer for rotated proposals.
RoIAwarePool3d	Encode the geometry-specific features of each 3D proposal.
RoIPointPool3d	Encode the geometry-specific features of each 3D proposal.
RoIPool	
SAConv2d	SAC (Switchable Atrous Convolution)
SigmoidFocalLoss	
SimpleRoIAlign	
SoftmaxFocalLoss	

下页继续

表 1 – 续上页

SparseConv2d	
SparseConv3d	
SparseConvTensor	
SparseConvTranspose2d	
SparseConvTranspose3d	
SparseInverseConv2d	
SparseInverseConv3d	
SparseMaxPool2d	
SparseMaxPool3d	
SparseModule	place holder, All module subclass from this will take sptensor in SparseSequential.
SparseSequential	A sequential container.
SubMConv2d	
SubMConv3d	
SyncBatchNorm	Synchronized Batch Normalization.
TINShift	Temporal Interlace Shift.
Voxelization	Convert kitti points(N, >=3) to voxels.

22.1 BorderAlign

class mmcv.ops.**BorderAlign**(*pool_size: int*)

Border align pooling layer.

Applies border_align over the input feature based on predicted bboxes. The details were described in the paper BorderDet: Border Feature for Dense Object Detection.

For each border line (e.g. top, left, bottom or right) of each box, border_align does the following:

1. uniformly samples `pool_size` +1 positions on this line, involving the start and end points.

2. the corresponding features on these points are computed by bilinear interpolation.
3. max pooling over all the `pool_size` +1 positions are used for computing pooled feature.

参数 **pool_size** (*int*) –number of positions sampled over the boxes' borders (e.g. top, bottom, left, right).

forward(*input: torch.Tensor*, *boxes: torch.Tensor*) → torch.Tensor

参数

- **input** –Features with shape [N,4C,H,W]. Channels ranged in [0,C), [C,2C), [2C,3C), [3C,4C) represent the top, left, bottom, right features respectively.
- **boxes** –Boxes with shape [N,H*W,4]. Coordinate format (x1,y1,x2,y2).

返回 Pooled features with shape [N,C,H*W,4]. The order is (top,left,bottom,right) for the last dimension.

返回类型 torch.Tensor

22.2 CARAFE

class mmcv.ops.**CARAFE**(*kernel_size: int*, *group_size: int*, *scale_factor: int*)

CARAFE: Content-Aware ReAssembly of FEatures

Please refer to CARAFE: Content-Aware ReAssembly of FEatures for more details.

参数

- **kernel_size** (*int*) –reassemble kernel size
- **group_size** (*int*) –reassemble group size
- **scale_factor** (*int*) –upsample ratio

返回 upsampled feature map

forward(*features: torch.Tensor*, *masks: torch.Tensor*) → torch.Tensor

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.

22.3 CARAFENaive

class mmcv.ops.**CARAFENaive**(*kernel_size: int*, *group_size: int*, *scale_factor: int*)

forward(*features: torch.Tensor*, *masks: torch.Tensor*) → torch.Tensor

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.

22.4 CARAFEPack

class mmcv.ops.**CARAFEPack**(*channels: int*, *scale_factor: int*, *up_kernel: int = 5*, *up_group: int = 1*, *encoder_kernel: int = 3*, *encoder_dilation: int = 1*, *compressed_channels: int = 64*)

A unified package of CARAFE upsampler that contains: 1) channel compressor 2) content encoder 3) CARAFE op.

Official implementation of ICCV 2019 paper CARAFE: Content-Aware ReAssembly of FEatures.

参数

- **channels** (*int*) –input feature channels
- **scale_factor** (*int*) –upsample ratio
- **up_kernel** (*int*) –kernel size of CARAFE op
- **up_group** (*int*) –group size of CARAFE op
- **encoder_kernel** (*int*) –kernel size of content encoder
- **encoder_dilation** (*int*) –dilation of content encoder
- **compressed_channels** (*int*) –output channels of channels compressor

返回 upsampled feature map

forward(*x: torch.Tensor*) → torch.Tensor

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.

22.5 Conv2d

`mmcv.ops.`**`Conv2d`**

alias of `mmcv.ops.deprecated_wrappers.Conv2d_deprecated`

22.6 ConvTranspose2d

`mmcv.ops.`**`ConvTranspose2d`**

alias of `mmcv.ops.deprecated_wrappers.ConvTranspose2d_deprecated`

22.7 CornerPool

class `mmcv.ops.`**`CornerPool`**(*mode: str*)

Corner Pooling.

Corner Pooling is a new type of pooling layer that helps a convolutional network better localize corners of bounding boxes.

Please refer to CornerNet: Detecting Objects as Paired Keypoints for more details.

Code is modified from https://github.com/princeton-vl/CornerNet-Lite.

参数 **mode** (*str*) –Pooling orientation for the pooling layer

- ’bottom’: Bottom Pooling
- ’left’: Left Pooling
- ’right’: Right Pooling
- ’top’: Top Pooling

返回 Feature map after pooling.

forward(*x: torch.Tensor*) → torch.Tensor

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.

22.8 Correlation

class mmcv.ops.**Correlation**(*kernel_size: int = 1*, *max_displacement: int = 1*, *stride: int = 1*, *padding: int = 0*, *dilation: int = 1*, *dilation_patch: int = 1*)

Correlation operator

This correlation operator works for optical flow correlation computation.

There are two batched tensors with shape (N, C, H, W), and the correlation output' s shape is $(N, max\_displacement \times 2 + 1, max\_displacement * 2 + 1, H_{out}, W_{out})$

where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1 \right\rfloor$$

the correlation item (N_i, dy, dx) is formed by taking the sliding window convolution between input1 and shifted input2,

$$Corr(N_i, dx, dy) = \sum_{c=0}^{C-1} input1(N_i, c) \star \mathcal{S}(input2(N_i, c), dy, dx)$$

where $\star$ is the valid 2d sliding window convolution operator, and $\mathcal{S}$ means shifting the input features (auto-complete zero marginal), and dx, dy are shifting distance, $dx, dy \in [-max\_displacement \times dilation\_patch, max\_displacement \times dilation\_patch]$.

参数

- **kernel_size** (*int*) –The size of sliding window i.e. local neighborhood representing the center points and involved in correlation computation. Defaults to 1.
- **max_displacement** (*int*) –The radius for computing correlation volume, but the actual working space can be dilated by dilation_patch. Defaults to 1.
- **stride** (*int*) –The stride of the sliding blocks in the input spatial dimensions. Defaults to 1.
- **padding** (*int*) –Zero padding added to all four sides of the input1. Defaults to 0.
- **dilation** (*int*) –The spacing of local neighborhood that will involved in correlation. Defaults to 1.

- **dilation_patch** (*int*) –The spacing between position need to compute correlation. Defaults to 1.

forward(*input1: torch.Tensor*, *input2: torch.Tensor*) → torch.Tensor

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.

22.9 CrissCrossAttention

class mmcv.ops.**CrissCrossAttention**(*in_channels: int*)

Criss-Cross Attention Module.

注解： Before v1.3.13, we use a CUDA op. Since v1.3.13, we switch to a pure PyTorch and equivalent implementation. For more details, please refer to https://github.com/open-mmlab/mmcv/pull/1201.

Speed comparison for one forward pass

- Input size: [2,512,97,97]
- Device: 1 NVIDIA GeForce RTX 2080 Ti

	PyTorch version	CUDA version	Relative speed
with torch.no_grad()	0.00554402 s	0.0299619 s	5.4x
no with torch.no_grad()	0.00562803 s	0.0301349 s	5.4x

参数 in_channels (*int*) –Channels of the input feature map.

forward(*x: torch.Tensor*) → torch.Tensor

forward function of Criss-Cross Attention.

参数 x (*torch.Tensor*) –Input feature with the shape of (batch_size, in_channels, height, width).

返回 Output of the layer, with the shape of (batch_size, in_channels, height, width)

返回类型 torch.Tensor

22.10 DeformConv2d

class mmcv.ops.**DeformConv2d**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, ...]]*, *stride: Union[int, Tuple[int, ...]] = 1*, *padding: Union[int, Tuple[int, ...]] = 0*, *dilation: Union[int, Tuple[int, ...]] = 1*, *groups: int = 1*, *deform_groups: int = 1*, *bias: bool = False*, *im2col_step: int = 32*)

Deformable 2D convolution.

Applies a deformable 2D convolution over an input signal composed of several input planes. DeformConv2d was described in the paper Deformable Convolutional Networks

---

注解： The argument `im2col_step` was added in version 1.3.17, which means number of samples processed by the `im2col_cuda_kernel` per call. It enables users to define `batch_size` and `im2col_step` more flexibly and solved issue mmcv#1440.

参数

- **in_channels** (*int*) –Number of channels in the input image.
- **out_channels** (*int*) –Number of channels produced by the convolution.
- **kernel_size** (*int, tuple*) –Size of the convolving kernel.
- **stride** (*int, tuple*) –Stride of the convolution. Default: 1.
- **padding** (*int or tuple*) –Zero-padding added to both sides of the input. Default: 0.
- **dilation** (*int or tuple*) –Spacing between kernel elements. Default: 1.
- **groups** (*int*) –Number of blocked connections from input. channels to output channels. Default: 1.
- **deform_groups** (*int*) –Number of deformable group partitions.
- **bias** (*bool*) –If True, adds a learnable bias to the output. Default: False.
- **im2col_step** (*int*) –Number of samples processed by im2col_cuda_kernel per call. It will work when `batch_size` > `im2col_step`, but `batch_size` must be divisible by `im2col_step`. Default: 32. *New in version 1.3.17.*

forward(*x: torch.Tensor*, *offset: torch.Tensor*) → torch.Tensor

Deformable Convolutional forward function.

参数

- **x** (*Tensor*) –Input feature, shape (B, C_in, H_in, W_in)

- **offset** (*Tensor*) –Offset for deformable convolution, shape (B, deform_groups*kernel_size[0]*kernel_size[1]*2, H_out, W_out), H_out, W_out are equal to the output's.

  An offset is like *[y0, x0, y1, x1, y2, x2, ⋯, y8, x8]*. The spatial arrangement is like:

```
(x0, y0) (x1, y1) (x2, y2)
(x3, y3) (x4, y4) (x5, y5)
(x6, y6) (x7, y7) (x8, y8)
```

返回 Output of the layer.

返回类型 Tensor

22.11 DeformConv2dPack

class mmcv.ops.**DeformConv2dPack**(**args*, ***kwargs*)

A Deformable Conv Encapsulation that acts as normal Conv layers.

The offset tensor is like *[y0, x0, y1, x1, y2, x2, ⋯, y8, x8]*. The spatial arrangement is like:

```
(x0, y0) (x1, y1) (x2, y2)
(x3, y3) (x4, y4) (x5, y5)
(x6, y6) (x7, y7) (x8, y8)
```

参数

- **in_channels** (*int*) –Same as nn.Conv2d.
- **out_channels** (*int*) –Same as nn.Conv2d.
- **kernel_size** (*int or tuple[int]*) –Same as nn.Conv2d.
- **stride** (*int or tuple[int]*) –Same as nn.Conv2d.
- **padding** (*int or tuple[int]*) –Same as nn.Conv2d.
- **dilation** (*int or tuple[int]*) –Same as nn.Conv2d.
- **groups** (*int*) –Same as nn.Conv2d.
- **bias** (*bool or str*) –If specified as *auto*, it will be decided by the norm_cfg. Bias will be set as True if norm_cfg is None, otherwise False.

forward(*x: torch.Tensor*) → torch.Tensor

Deformable Convolutional forward function.

参数

- **x** (*Tensor*) –Input feature, shape (B, C_in, H_in, W_in)

- **offset** (*Tensor*) –Offset for deformable convolution, shape (B, deform_groups*kernel_size[0]*kernel_size[1]*2, H_out, W_out), H_out, W_out are equal to the output’s.

  An offset is like *[y0, x0, y1, x1, y2, x2, ⋯, y8, x8]*. The spatial arrangement is like:

```
(x0, y0) (x1, y1) (x2, y2)
(x3, y3) (x4, y4) (x5, y5)
(x6, y6) (x7, y7) (x8, y8)
```

返回 Output of the layer.

返回类型 Tensor

22.12 DeformRoIPool

class mmcv.ops.**DeformRoIPool**(*output_size: Tuple[int, ...]*, *spatial_scale: float = 1.0*, *sampling_ratio: int = 0*, *gamma: float = 0.1*)

forward(*input: torch.Tensor*, *rois: torch.Tensor*, *offset: Optional[torch.Tensor] = None*) → torch.Tensor

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.

22.13 DeformRoIPoolPack

class mmcv.ops.**DeformRoIPoolPack**(*output_size: Tuple[int, ...]*, *output_channels: int*, *deform_fc_channels: int = 1024*, *spatial_scale: float = 1.0*, *sampling_ratio: int = 0*, *gamma: float = 0.1*)

forward(*input: torch.Tensor*, *rois: torch.Tensor*) → torch.Tensor

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.

22.14 DynamicScatter

class mmcv.ops.**DynamicScatter**(*voxel_size: List*, *point_cloud_range: List*, *average_points: bool*)

Scatters points into voxels, used in the voxel encoder with dynamic voxelization.

注解： The CPU and GPU implementation get the same output, but have numerical difference after summation and division (e.g., 5e-7).

参数

- **voxel_size** (*list*) –list [x, y, z] size of three dimension.
- **point_cloud_range** (*list*) –The coordinate range of points, [x_min, y_min, z_min, x_max, y_max, z_max].
- **average_points** (*bool*) –whether to use avg pooling to scatter points into voxel.

forward(*points: torch.Tensor*, *coors: torch.Tensor*) → Tuple[torch.Tensor, torch.Tensor]

Scatters points/features into voxels.

参数

- **points** (*torch.Tensor*) –Points to be reduced into voxels.
- **coors** (*torch.Tensor*) –Corresponding voxel coordinates (specifically multi-dim voxel index) of each points.

返回 A tuple contains two elements. The first one is the voxel features with shape [M, C] which are respectively reduced from input features that share the same voxel coordinates. The second is voxel coordinates with shape [M, ndim].

返回类型 tuple[torch.Tensor]

forward_single(*points: torch.Tensor*, *coors: torch.Tensor*) → Tuple[torch.Tensor, torch.Tensor]

Scatters points into voxels.

参数

- **points** (*torch.Tensor*) –Points to be reduced into voxels.
- **coors** (*torch.Tensor*) –Corresponding voxel coordinates (specifically multi-dim voxel index) of each points.

返回 A tuple contains two elements. The first one is the voxel features with shape [M, C] which are respectively reduced from input features that share the same voxel coordinates. The second is voxel coordinates with shape [M, ndim].

返回类型 tuple[torch.Tensor]

22.15 FusedBiasLeakyReLU

class mmcv.ops.**FusedBiasLeakyReLU**(*num_channels: int*, *negative_slope: float = 0.2*, *scale: float = 1.4142135623730951*)

Fused bias leaky ReLU.

This function is introduced in the StyleGAN2: Analyzing and Improving the Image Quality of StyleGAN

The bias term comes from the convolution operation. In addition, to keep the variance of the feature map or gradients unchanged, they also adopt a scale similarly with Kaiming initialization. However, since the $1 + alpha^2$ is too small, we can just ignore it. Therefore, the final scale is just $\sqrt{2}$. Of course, you may change it with your own scale.

TODO: Implement the CPU version.

参数

- **num_channels** (*int*) –The channel number of the feature map.
- **negative_slope** (*float, optional*) –Same as nn.LeakyRelu. Defaults to 0.2.
- **scale** (*float, optional*) –A scalar to adjust the variance of the feature map. Defaults to 2**0.5.

forward(*input: torch.Tensor*) → torch.Tensor

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.

22.16 GroupAll

class mmcv.ops.**GroupAll**(*use_xyz: bool = True*)

Group xyz with feature.

参数 **use_xyz** (*bool*) –Whether to use xyz.

forward(*xyz: torch.Tensor*, *new_xyz: torch.Tensor*, *features: Optional[torch.Tensor] = None*) → torch.Tensor

参数

- **xyz** (*Tensor*) –(B, N, 3) xyz coordinates of the features.
- **new_xyz** (*Tensor*) –new xyz coordinates of the features.
- **features** (*Tensor*) –(B, C, N) features to group.

返回 (B, C + 3, 1, N) Grouped feature.

返回类型 Tensor

22.17 Linear

mmcv.ops.**Linear**

alias of mmcv.ops.deprecated_wrappers.Linear_deprecated

22.18 MaskedConv2d

class mmcv.ops.**MaskedConv2d**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int, ...]]*, *stride: int = 1*, *padding: int = 0*, *dilation: int = 1*, *groups: int = 1*, *bias: bool = True*)

A MaskedConv2d which inherits the official Conv2d.

The masked forward doesn't implement the backward function and only supports the stride parameter to be 1 currently.

forward(*input: torch.Tensor*, *mask: Optional[torch.Tensor] = None*) → torch.Tensor

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.

22.19 MaxPool2d

mmcv.ops.**MaxPool2d**

alias of mmcv.ops.deprecated_wrappers.MaxPool2d_deprecated

22.20 ModulatedDeformConv2d

class mmcv.ops.**ModulatedDeformConv2d**(*in_channels: int*, *out_channels: int*, *kernel_size: Union[int, Tuple[int]]*, *stride: int = 1*, *padding: int = 0*, *dilation: int = 1*, *groups: int = 1*, *deform_groups: int = 1*, *bias: Union[bool, str] = True*)

forward(*x: torch.Tensor*, *offset: torch.Tensor*, *mask: torch.Tensor*) → torch.Tensor

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.

22.21 ModulatedDeformConv2dPack

class mmcv.ops.**ModulatedDeformConv2dPack**(**args*, ***kwargs*)

A ModulatedDeformable Conv Encapsulation that acts as normal Conv layers.

参数

- **in_channels** (*int*) –Same as nn.Conv2d.
- **out_channels** (*int*) –Same as nn.Conv2d.
- **kernel_size** (*int or tuple[int]*) –Same as nn.Conv2d.
- **stride** (*int*) –Same as nn.Conv2d, while tuple is not supported.
- **padding** (*int*) –Same as nn.Conv2d, while tuple is not supported.
- **dilation** (*int*) –Same as nn.Conv2d, while tuple is not supported.

- **groups** (*int*) –Same as nn.Conv2d.
- **bias** (*bool or str*) –If specified as *auto*, it will be decided by the norm_cfg. Bias will be set as True if norm_cfg is None, otherwise False.

forward(*x: torch.Tensor*) → torch.Tensor

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.

22.22 ModulatedDeformRoIPoolPack

class mmcv.ops.**ModulatedDeformRoIPoolPack**(*output_size: Tuple[int, ...]*, *output_channels: int*, *deform_fc_channels: int = 1024*, *spatial_scale: float = 1.0*, *sampling_ratio: int = 0*, *gamma: float = 0.1*)

forward(*input: torch.Tensor*, *rois: torch.Tensor*) → torch.Tensor

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.

22.23 MultiScaleDeformableAttention

class mmcv.ops.**MultiScaleDeformableAttention**(*embed_dims: int = 256*, *num_heads: int = 8*, *num_levels: int = 4*, *num_points: int = 4*, *im2col_step: int = 64*, *dropout: float = 0.1*, *batch_first: bool = False*, *norm_cfg: Optional[dict] = None*, *init_cfg: Optional[mmengine.config.config.ConfigDict] = None*, *value_proj_ratio: float = 1.0*)

An attention module used in Deformable-Detr.

Deformable DETR: Deformable Transformers for End-to-End Object Detection..

参数

- **embed_dims** (*int*) –The embedding dimension of Attention. Default: 256.
- **num_heads** (*int*) –Parallel attention heads. Default: 8.
- **num_levels** (*int*) –The number of feature map used in Attention. Default: 4.
- **num_points** (*int*) –The number of sampling points for each query in each head. Default: 4.
- **im2col_step** (*int*) –The step used in image_to_column. Default: 64.
- **dropout** (*float*) –A Dropout layer on *inp_identity*. Default: 0.1.
- **batch_first** (*bool*) –Key, Query and Value are shape of (batch, n, embed_dim) or (n, batch, embed_dim). Default to False.
- **norm_cfg** (*dict*) –Config dict for normalization layer. Default: None.
- **(obj** (*init_cfg*) –*mmcv.ConfigDict*): The Config for initialization. Default: None.
- **value_proj_ratio** (*float*) –The expansion ratio of value_proj. Default: 1.0.

forward(*query: torch.Tensor*, *key: Optional[torch.Tensor] = None*, *value: Optional[torch.Tensor] = None*, *identity: Optional[torch.Tensor] = None*, *query_pos: Optional[torch.Tensor] = None*, *key_padding_mask: Optional[torch.Tensor] = None*, *reference_points: Optional[torch.Tensor] = None*, *spatial_shapes: Optional[torch.Tensor] = None*, *level_start_index: Optional[torch.Tensor] = None*, ***kwargs*) → torch.Tensor

Forward Function of MultiScaleDeformAttention.

参数

- **query** (*torch.Tensor*) –Query of Transformer with shape (num_query, bs, embed_dims).
- **key** (*torch.Tensor*) –The key tensor with shape *(num_key, bs, embed_dims)*.
- **value** (*torch.Tensor*) –The value tensor with shape *(num_key, bs, embed_dims)*.
- **identity** (*torch.Tensor*) –The tensor used for addition, with the same shape as *query*. Default None. If None, *query* will be used.
- **query_pos** (*torch.Tensor*) –The positional encoding for *query*. Default: None.
- **key_padding_mask** (*torch.Tensor*) –ByteTensor for *query*, with shape [bs, num_key].
- **reference_points** (*torch.Tensor*) –The normalized reference points with shape (bs, num_query, num_levels, 2), all elements is range in [0, 1], top-left (0,0), bottom-right (1, 1), including padding area. or (N, Length_{query}, num_levels, 4), add additional two dimensions is (w, h) to form reference boxes.

- **spatial_shapes** (*torch.Tensor*) –Spatial shape of features in different levels. With shape (num_levels, 2), last dimension represents (h, w).
- **level_start_index** (*torch.Tensor*) –The start index of each level. A tensor has shape `(num_levels, )` and can be represented as [0, h_0*w_0, h_0*w_0+h_1*w_1, ···].

返回 forwarded results with shape [num_query, bs, embed_dims].

返回类型 torch.Tensor

init_weights() → None

Default initialization for Parameters of Module.

22.24 PSAMask

class mmcv.ops.**PSAMask**(*psa_type: str*, *mask_size: Optional[tuple] = None*)

forward(*input: torch.Tensor*) → torch.Tensor

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.

22.25 PointsSampler

class mmcv.ops.**PointsSampler**(*num_point: List[int]*, *fps_mod_list: List[str] = ['D-FPS']*, *fps_sample_range_list: List[int] = [- 1]*)

Points sampling.

参数

- **num_point** (*list[int]*) –Number of sample points.
- **fps_mod_list** (*list[str], optional*) –Type of FPS method, valid mod ['F-FPS' , 'D-FPS' , 'FS'], Default: ['D-FPS']. F-FPS: using feature distances for FPS. D-FPS: using Euclidean distances of points for FPS. FS: using F-FPS and D-FPS simultaneously.
- **fps_sample_range_list** (*list[int], optional*) –Range of points to apply FPS. Default: [-1].

forward(*points_xyz: torch.Tensor*, *features: torch.Tensor*) → torch.Tensor

参数

- **points_xyz** (*torch.Tensor*) –(B, N, 3) xyz coordinates of the points.
- **features** (*torch.Tensor*) –(B, C, N) features of the points.

返回 (B, npoint, sample_num) Indices of sampled points.

返回类型 torch.Tensor

22.26 PrRoIPool

class mmcv.ops.**PrRoIPool**(*output_size: Union[int, tuple]*, *spatial_scale: float = 1.0*)

The operation of precision RoI pooling. The implementation of PrRoIPool is modified from https://github.com/vacancy/PreciseRoIPooling/

Precise RoI Pooling (PrRoIPool) is an integration-based (bilinear interpolation) average pooling method for RoI Pooling. It avoids any quantization and has a continuous gradient on bounding box coordinates. It is:

1. different from the original RoI Pooling proposed in Fast R-CNN. PrRoI Pooling uses average pooling instead of max pooling for each bin and has a continuous gradient on bounding box coordinates. That is, one can take the derivatives of some loss function w.r.t the coordinates of each RoI and optimize the RoI coordinates. 2. different from the RoI Align proposed in Mask R-CNN. PrRoI Pooling uses a full integration-based average pooling instead of sampling a constant number of points. This makes the gradient w.r.t. the coordinates continuous.

参数

- **output_size** (*Union[int, tuple]*) –h, w.
- **spatial_scale** (*float, optional*) –scale the input boxes by this number. Defaults to 1.0.

forward(*features: torch.Tensor*, *rois: torch.Tensor*) → torch.Tensor

Forward function.

参数

- **features** (*torch.Tensor*) –The feature map.
- **rois** (*torch.Tensor*) –The RoI bboxes in [tl_x, tl_y, br_x, br_y] format.

返回 The pooled results.

返回类型 torch.Tensor

22.27 QueryAndGroup

class mmcv.ops.**QueryAndGroup**(*max_radius: float*, *sample_num: int*, *min_radius: float = 0.0*, *use_xyz: bool = True*, *return_grouped_xyz: bool = False*, *normalize_xyz: bool = False*, *uniform_sample: bool = False*, *return_unique_cnt: bool = False*, *return_grouped_idx: bool = False*)

Groups points with a ball query of radius.

参数

- **max_radius** (*float*) –The maximum radius of the balls. If None is given, we will use kNN sampling instead of ball query.
- **sample_num** (*int*) –Maximum number of features to gather in the ball.
- **min_radius** (*float, optional*) –The minimum radius of the balls. Default: 0.
- **use_xyz** (*bool, optional*) –Whether to use xyz. Default: True.
- **return_grouped_xyz** (*bool, optional*) –Whether to return grouped xyz. Default: False.
- **normalize_xyz** (*bool, optional*) –Whether to normalize xyz. Default: False.
- **uniform_sample** (*bool, optional*) –Whether to sample uniformly. Default: False
- **return_unique_cnt** (*bool, optional*) –Whether to return the count of unique samples. Default: False.
- **return_grouped_idx** (*bool, optional*) –Whether to return grouped idx. Default: False.

forward(*points_xyz: torch.Tensor*, *center_xyz: torch.Tensor*, *features: Optional[torch.Tensor] = None*) → Union[torch.Tensor, Tuple]

参数

- **points_xyz** (*torch.Tensor*) –(B, N, 3) xyz coordinates of the points.
- **center_xyz** (*torch.Tensor*) –(B, npoint, 3) coordinates of the centriods.
- **features** (*torch.Tensor*) –(B, C, N) The features of grouped points.

返回 (B, 3 + C, npoint, sample_num) Grouped concatenated coordinates and features of points.

返回类型 Tuple | torch.Tensor

22.28 RiRoIAlignRotated

class mmcv.ops.**RiRoIAlignRotated**(*out_size: tuple*, *spatial_scale: float*, *num_samples: int = 0*, *num_orientations: int = 8*, *clockwise: bool = False*)

Rotation-invariant RoI align pooling layer for rotated proposals.

It accepts a feature map of shape (N, C, H, W) and rois with shape (n, 6) with each roi decoded as (batch_index, center_x, center_y, w, h, angle). The angle is in radian.

The details are described in the paper ReDet: A Rotation-equivariant Detector for Aerial Object Detection.

参数

- **out_size** (*tuple*) –fixed dimensional RoI output with shape (h, w).
- **spatial_scale** (*float*) –scale the input boxes by this number
- **num_samples** (*int*) –number of inputs samples to take for each output sample. 0 to take samples densely for current models.
- **num_orientations** (*int*) –number of oriented channels.
- **clockwise** (*bool*) –If True, the angle in each proposal follows a clockwise fashion in image space, otherwise, the angle is counterclockwise. Default: False.

forward(*features: torch.Tensor*, *rois: torch.Tensor*) → torch.Tensor

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.

22.29 RoIAlign

class mmcv.ops.**RoIAlign**(*output_size: tuple*, *spatial_scale: float = 1.0*, *sampling_ratio: int = 0*, *pool_mode: str = 'avg'*, *aligned: bool = True*, *use_torchvision: bool = False*)

RoI align pooling layer.

参数

- **output_size** (*tuple*) –h, w
- **spatial_scale** (*float*) –scale the input boxes by this number

- **sampling_ratio** (*int*) –number of inputs samples to take for each output sample. 0 to take samples densely for current models.
- **pool_mode** (*str, 'avg' or 'max'*) –pooling mode in each bin.
- **aligned** (*bool*) –if False, use the legacy implementation in MMDetection. If True, align the results more perfectly.
- **use_torchvision** (*bool*) –whether to use roi_align from torchvision.

注解: The implementation of RoIAlign when aligned=True is modified from https://github.com/facebookresearch/detectron2/

The meaning of aligned=True:

Given a continuous coordinate c, its two neighboring pixel indices (in our pixel model) are computed by floor(c - 0.5) and ceil(c - 0.5). For example, c=1.3 has pixel neighbors with discrete indices [0] and [1] (which are sampled from the underlying signal at continuous coordinates 0.5 and 1.5). But the original roi_align (aligned=False) does not subtract the 0.5 when computing neighboring pixel indices and therefore it uses pixels with a slightly incorrect alignment (relative to our pixel model) when performing bilinear interpolation.

With *aligned=True*, we first appropriately scale the ROI and then shift it by -0.5 prior to calling roi_align. This produces the correct neighbors;

The difference does not make a difference to the model's performance if ROIAlign is used together with conv layers.

---

forward(*input: torch.Tensor*, *rois: torch.Tensor*) → torch.Tensor

参数

- **input** –NCHW images
- **rois** –Bx5 boxes. First column is the index into N. The other 4 columns are xyxy.

22.30 RoIAlignRotated

class mmcv.ops.**RoIAlignRotated**(*output_size: Union[int, tuple]*, *spatial_scale: float*, *sampling_ratio: int = 0*, *aligned: bool = True*, *clockwise: bool = False*)

RoI align pooling layer for rotated proposals.

It accepts a feature map of shape (N, C, H, W) and rois with shape (n, 6) with each roi decoded as (batch_index, center_x, center_y, w, h, angle). The angle is in radian.

参数

- **output_size** (*tuple*) –h, w

- **spatial_scale** (*float*) –scale the input boxes by this number
- **sampling_ratio** (*int*) –number of inputs samples to take for each output sample. 0 to take samples densely for current models.
- **aligned** (*bool*) –if False, use the legacy implementation in MMDetection. If True, align the results more perfectly. Default: True.
- **clockwise** (*bool*) –If True, the angle in each proposal follows a clockwise fashion in image space, otherwise, the angle is counterclockwise. Default: False.

注解: The implementation of RoIAlign when aligned=True is modified from https://github.com/facebookresearch/detectron2/

The meaning of aligned=True:

Given a continuous coordinate c, its two neighboring pixel indices (in our pixel model) are computed by floor(c - 0.5) and ceil(c - 0.5). For example, c=1.3 has pixel neighbors with discrete indices [0] and [1] (which are sampled from the underlying signal at continuous coordinates 0.5 and 1.5). But the original roi_align (aligned=False) does not subtract the 0.5 when computing neighboring pixel indices and therefore it uses pixels with a slightly incorrect alignment (relative to our pixel model) when performing bilinear interpolation.

With *aligned=True*, we first appropriately scale the ROI and then shift it by -0.5 prior to calling roi_align. This produces the correct neighbors;

The difference does not make a difference to the model's performance if ROIAlign is used together with conv layers.

---

forward(*input: torch.Tensor*, *rois: torch.Tensor*) → torch.Tensor

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.

22.31 RoIAwarePool3d

class mmcv.ops.**RoIAwarePool3d**(*out_size: Union[int, tuple]*, *max_pts_per_voxel: int = 128*, *mode: str = 'max'*)

Encode the geometry-specific features of each 3D proposal.

Please refer to PartA2 for more details.

参数

- **out_size** (*int or tuple*) –The size of output features. n or [n1, n2, n3].
- **max_pts_per_voxel** (*int, optional*) –The maximum number of points per voxel. Default: 128.
- **mode** (*str, optional*) –Pooling method of RoIAware, 'max' or 'avg'. Default: 'max'.

forward(*rois: torch.Tensor*, *pts: torch.Tensor*, *pts_feature: torch.Tensor*) → torch.Tensor

参数

- **rois** (*torch.Tensor*) –[N, 7], in LiDAR coordinate, (x, y, z) is the bottom center of rois.
- **pts** (*torch.Tensor*) –[npoints, 3], coordinates of input points.
- **pts_feature** (*torch.Tensor*) –[npoints, C], features of input points.

返回 Pooled features whose shape is [N, out_x, out_y, out_z, C].

返回类型 torch.Tensor

22.32 RoIPointPool3d

class mmcv.ops.**RoIPointPool3d**(*num_sampled_points: int = 512*)

Encode the geometry-specific features of each 3D proposal.

Please refer to Paper of PartA2 for more details.

参数 **num_sampled_points** (*int, optional*) –Number of samples in each roi. Default: 512.

forward(*points: torch.Tensor*, *point_features: torch.Tensor*, *boxes3d: torch.Tensor*) → Tuple[torch.Tensor]

参数

- **points** (*torch.Tensor*) –Input points whose shape is (B, N, C).

- **point_features** (*torch.Tensor*) –Features of input points whose shape is (B, N, C).
- **boxes3d** (*B, M, 7), Input bounding boxes whose shape is (B, M, 7*) –

返回 A tuple contains two elements. The first one is the pooled features whose shape is (B, M, 512, 3 + C). The second is an empty flag whose shape is (B, M).

返回类型 tuple[torch.Tensor]

22.33 RoIPool

class mmcv.ops.**RoIPool**(*output_size: Union[int, tuple]*, *spatial_scale: float = 1.0*)

forward(*input: torch.Tensor*, *rois: torch.Tensor*) → torch.Tensor

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.

22.34 SAConv2d

class mmcv.ops.**SAConv2d**(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *use_deform=False*)

SAC (Switchable Atrous Convolution)

This is an implementation of DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution.

参数

- **in_channels** (*int*) –Number of channels in the input image
- **out_channels** (*int*) –Number of channels produced by the convolution
- **kernel_size** (*int or tuple*) –Size of the convolving kernel
- **stride** (*int or tuple, optional*) –Stride of the convolution. Default: 1
- **padding** (*int or tuple, optional*) –Zero-padding added to both sides of the input. Default: 0

- **padding_mode** (*string, optional*) – `'zeros'`, `'reflect'`, `'replicate'` or `'circular'`. Default: `'zeros'`
- **dilation** (*int or tuple, optional*) – Spacing between kernel elements. Default: 1
- **groups** (*int, optional*) – Number of blocked connections from input channels to output channels. Default: 1
- **bias** (*bool, optional*) – If `True`, adds a learnable bias to the output. Default: `True`
- **use_deform** – If `True`, replace convolution with deformable convolution. Default: `False`.

forward(*x*)

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.

22.35 SigmoidFocalLoss

class mmcv.ops.**SigmoidFocalLoss**(*gamma: float*, *alpha: float*, *weight: Optional[torch.Tensor] = None*, *reduction: str = 'mean'*)

forward(*input: torch.Tensor*, *target: Union[torch.LongTensor, torch.cuda.LongTensor]*) → torch.Tensor

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.

22.36 SimpleRoIAlign

class mmcv.ops.**SimpleRoIAlign**(*output_size: Tuple[int]*, *spatial_scale: float*, *aligned: bool = True*)

forward(*features: torch.Tensor*, *rois: torch.Tensor*) → torch.Tensor

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.

22.37 SoftmaxFocalLoss

class mmcv.ops.**SoftmaxFocalLoss**(*gamma: float*, *alpha: float*, *weight: Optional[torch.Tensor] = None*, *reduction: str = 'mean'*)

forward(*input: torch.Tensor*, *target: Union[torch.LongTensor, torch.cuda.LongTensor]*) → torch.Tensor

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.

22.38 SparseConv2d

class mmcv.ops.**SparseConv2d**(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *indice_key=None*)

22.39 SparseConv3d

class mmcv.ops.**SparseConv3d**(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *indice_key=None*)

22.40 SparseConvTensor

class mmcv.ops.**SparseConvTensor**(*features: torch.Tensor*, *indices: torch.Tensor*, *spatial_shape: Union[List, Tuple]*, *batch_size: int*, *grid: Optional[torch.Tensor] = None*)

22.41 SparseConvTranspose2d

class mmcv.ops.**SparseConvTranspose2d**(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *indice_key=None*)

22.42 SparseConvTranspose3d

class mmcv.ops.**SparseConvTranspose3d**(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *indice_key=None*)

22.43 SparseInverseConv2d

class mmcv.ops.**SparseInverseConv2d**(*in_channels*, *out_channels*, *kernel_size*, *indice_key=None*, *bias=True*)

22.44 SparseInverseConv3d

class mmcv.ops.**SparseInverseConv3d**(*in_channels*, *out_channels*, *kernel_size*, *indice_key=None*, *bias=True*)

22.45 SparseMaxPool2d

class mmcv.ops.**SparseMaxPool2d**(*kernel_size*, *stride=1*, *padding=0*, *dilation=1*)

22.46 SparseMaxPool3d

class mmcv.ops.**SparseMaxPool3d**(*kernel_size*, *stride=1*, *padding=0*, *dilation=1*)

22.47 SparseModule

class mmcv.ops.**SparseModule**

place holder, All module subclass from this will take sptensor in SparseSequential.

22.48 SparseSequential

class mmcv.ops.**SparseSequential**(**args*, ***kwargs*)

A sequential container. Modules will be added to it in the order they are passed in the constructor. Alternatively, an ordered dict of modules can also be passed in.

To make it easier to understand, given is a small example:

.. rubric:: 示例

```
>>> # using Sequential:
>>> from mmcv.ops import SparseSequential
>>> model = SparseSequential(
            SparseConv2d(1,20,5),
            nn.ReLU(),
            SparseConv2d(20,64,5),
            nn.ReLU()
            )
```

```
>>> # using Sequential with OrderedDict
>>> model = SparseSequential(OrderedDict([
              ('conv1', SparseConv2d(1,20,5)),
              ('relu1', nn.ReLU()),
              ('conv2', SparseConv2d(20,64,5)),
              ('relu2', nn.ReLU())
            ]))
```

```
>>> # using Sequential with kwargs(python 3.6+)
>>> model = SparseSequential(
              conv1=SparseConv2d(1,20,5),
              relu1=nn.ReLU(),
              conv2=SparseConv2d(20,64,5),
              relu2=nn.ReLU()
            )
```

forward(*input: torch.Tensor*) → torch.Tensor

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.

22.49 SubMConv2d

class mmcv.ops.**SubMConv2d**(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *indice_key=None*)

22.50 SubMConv3d

class mmcv.ops.**SubMConv3d**(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *indice_key=None*)

22.51 SyncBatchNorm

class mmcv.ops.**SyncBatchNorm**(*num_features: int*, *eps: float = 1e-05*, *momentum: float = 0.1*, *affine: bool = True*, *track_running_stats: bool = True*, *group: Optional[int] = None*, *stats_mode: str = 'default'*)

Synchronized Batch Normalization.

参数

- **num_features** (*int*) –number of features/chennels in input tensor
- **eps** (*float, optional*) –a value added to the denominator for numerical stability. Defaults to 1e-5.

- **momentum** (*float, optional*) –the value used for the running_mean and running_var computation. Defaults to 0.1.
- **affine** (*bool, optional*) –whether to use learnable affine parameters. Defaults to True.
- **track_running_stats** (*bool, optional*) –whether to track the running mean and variance during training. When set to False, this module does not track such statistics, and initializes statistics buffers `running_mean` and `running_var` as `None`. When these buffers are `None`, this module always uses batch statistics in both training and eval modes. Defaults to True.
- **group** (*int, optional*) –synchronization of stats happen within each process group individually. By default it is synchronization across the whole world. Defaults to None.
- **stats_mode** (*str, optional*) –The statistical mode. Available options includes `'default'` and `'N'`. Defaults to 'default'. When `stats_mode=='default'`, it computes the overall statistics using those from each worker with equal weight, i.e., the statistics are synchronized and simply divied by `group`. This mode will produce inaccurate statistics when empty tensors occur. When `stats_mode=='N'`, it compute the overall statistics using the total number of batches in each worker ignoring the number of group, i.e., the statistics are synchronized and then divied by the total batch `N`. This mode is beneficial when empty tensors occur during training, as it average the total mean by the real number of batch.

forward(*input: torch.Tensor*) → torch.Tensor

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.

22.52 TINShift

class mmcv.ops.**TINShift**

Temporal Interlace Shift.

Temporal Interlace shift is a differentiable temporal-wise frame shifting which is proposed in "Temporal Interlacing Network"

Please refer to Temporal Interlacing Network for more details.

Code is modified from https://github.com/mit-han-lab/temporal-shift-module

forward(*input*, *shift*)

Perform temporal interlace shift.

参数

- **input** (*torch.Tensor*) –Feature map with shape [N, num_segments, C, H * W].
- **shift** (*torch.Tensor*) –Shift tensor with shape [N, num_segments].

返回 Feature map after temporal interlace shift.

22.53 Voxelization

class mmcv.ops.**Voxelization**(*voxel_size: List*, *point_cloud_range: List*, *max_num_points: int*, *max_voxels: Union[tuple, int] = 20000*, *deterministic: bool = True*)

Convert kitti points(N, >=3) to voxels.

Please refer to Point-Voxel CNN for Efficient 3D Deep Learning for more details.

参数

- **voxel_size** (*tuple or float*) –The size of voxel with the shape of [3].
- **point_cloud_range** (*tuple or float*) –The coordinate range of voxel with the shape of [6].
- **max_num_points** (*int*) –maximum points contained in a voxel. if max_points=-1, it means using dynamic_voxelize.
- **max_voxels** (*int, optional*) –maximum voxels this function create. for second, 20000 is a good choice. Users should shuffle points before call this function because max_voxels may drop points. Default: 20000.

forward(*input: torch.Tensor*) → torch.Tensor

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.

---

active_rotated_filter

下页继续

表 2 – 续上页

assign_score_withk	
ball_query	
batched_nms	Performs non-maximum suppression in a batched fashion.
bbox_overlaps	Calculate overlap between two set of bboxes.
border_align	
box_iou_rotated	Return intersection-over-union (Jaccard index) of boxes.
boxes_iou3d	Calculate boxes 3D IoU.
boxes_iou_bev	Calculate boxes IoU in the Bird’s Eye View.
boxes_overlap_bev	Calculate boxes BEV overlap.
carafe	
carafe_naive	
chamfer_distance	
contour_expand	Expand kernel contours so that foreground pixels are assigned into instances.
convex_giou	Return generalized intersection-over-union (Jaccard index) between point sets and polygons.
convex_iou	Return intersection-over-union (Jaccard index) between point sets and polygons.
deform_conv2d	
deform_roi_pool	
diff_iou_rotated_2d	Calculate differentiable iou of rotated 2d boxes.
diff_iou_rotated_3d	Calculate differentiable iou of rotated 3d boxes.
dynamic_scatter	
furthest_point_sample	
furthest_point_sample_with_dist	
fused_bias_leakyrelu	Fused bias leaky ReLU function.
gather_points	

下页继续

表 2 – 续上页

grouping_operation	
knn	
masked_conv2d	
min_area_polygons	Find the smallest polygons that surrounds all points in the point sets.
modulated_deform_conv2d	
nms	Dispatch to either CPU or GPU NMS implementations.
nms3d	3D NMS function GPU implementation (for BEV boxes).
nms3d_normal	Normal 3D NMS function GPU implementation.
nms_bev	NMS function GPU implementation (for BEV boxes).
nms_match	Matched dets into different groups by NMS.
nms_normal_bev	Normal NMS function GPU implementation (for BEV boxes).
nms_rotated	Performs non-maximum suppression (NMS) on the rotated boxes according to their intersection-over-union (IoU).
pixel_group	Group pixels into text instances, which is widely used text detection methods.
point_sample	A wrapper around `grid_sample()` to support 3D point_coords tensors Unlike `torch.nn.functional.grid_sample()` it assumes point_coords to lie inside `[0, 1] x [0, 1]` square.
points_in_boxes_all	Find all boxes in which each point is (CUDA).
points_in_boxes_cpu	Find all boxes in which each point is (CPU).
points_in_boxes_part	Find the box in which each point is (CUDA).
points_in_polygons	Judging whether points are inside polygons, which is used in the ATSS assignment for the rotated boxes.
prroi_pool	
rel_roi_point_to_rel_img_point	Convert roi based relative point coordinates to image based absolute point coordinates.
riroi_align_rotated	

下页继续

表 2 – 续上页

roi_align	
roi_align_rotated	
roi_pool	
rotated_feature_align	
scatter_nd	pytorch edition of tensorflow scatter_nd.
sigmoid_focal_loss	
soft_nms	Dispatch to only CPU Soft NMS implementations.
softmax_focal_loss	
three_interpolate	
three_nn	
tin_shift	
upfirdn2d	Pad, upsample, filter, and downsample a batch of 2D images.
voxelization	

22.54 mmcv.ops.active_rotated_filter

mmcv.ops.**active_rotated_filter**()

22.55 mmcv.ops.assign_score_withk

mmcv.ops.**assign_score_withk**()

22.56 mmcv.ops.ball_query

mmcv.ops.**ball_query**()

22.57 mmcv.ops.batched_nms

mmcv.ops.**batched_nms**(*boxes: torch.Tensor*, *scores: torch.Tensor*, *idxs: torch.Tensor*, *nms_cfg: Optional[Dict]*, *class_agnostic: bool = False*) → Tuple[torch.Tensor, torch.Tensor]

Performs non-maximum suppression in a batched fashion.

Modified from torchvision/ops/boxes.py#L39. In order to perform NMS independently per class, we add an offset to all the boxes. The offset is dependent only on the class idx, and is large enough so that boxes from different classes do not overlap.

注解: In v1.4.1 and later, `batched_nms` supports skipping the NMS and returns sorted raw results when *nms_cfg* is None.

参数

- **boxes** (*torch.Tensor*) –boxes in shape (N, 4) or (N, 5).
- **scores** (*torch.Tensor*) –scores in shape (N,).
- **idxs** (*torch.Tensor*) –each index value correspond to a bbox cluster, and NMS will not be applied between elements of different idxs, shape (N,).
- **nms_cfg** (*dict | optional*) –Supports skipping the nms when *nms_cfg* is None, otherwise it should specify nms type and other parameters like *iou_thr*. Possible keys includes the following.
  - iou_threshold (float): IoU threshold used for NMS.
  - split_thr (float): threshold number of boxes. In some cases the number of boxes is large (e.g., 200k). To avoid OOM during training, the users could set *split_thr* to a small value. If the number of boxes is greater than the threshold, it will perform NMS on each group of boxes separately and sequentially. Defaults to 10000.
- **class_agnostic** (*bool*) –if true, nms is class agnostic, i.e. IoU thresholding happens over all boxes, regardless of the predicted class. Defaults to False.

返回

kept dets and indice.

- boxes (Tensor): Bboxes with score after nms, has shape (num_bboxes, 5). last dimension 5 arrange as (x1, y1, x2, y2, score)
- keep (Tensor): The indices of remaining boxes in input boxes.

返回类型 tuple

22.58 mmcv.ops.bbox_overlaps

mmcv.ops.**bbox_overlaps**(*bboxes1: torch.Tensor*, *bboxes2: torch.Tensor*, *mode: str = 'iou'*, *aligned: bool = False*, *offset: int = 0*) → torch.Tensor

Calculate overlap between two set of bboxes.

If `aligned` is `False`, then calculate the ious between each bbox of bboxes1 and bboxes2, otherwise the ious between each aligned pair of bboxes1 and bboxes2.

参数

- **bboxes1** (*torch.Tensor*) –shape (m, 4) in <x1, y1, x2, y2> format or empty.
- **bboxes2** (*torch.Tensor*) –shape (n, 4) in <x1, y1, x2, y2> format or empty. If aligned is `True`, then m and n must be equal.
- **mode** (*str*) – "iou" (intersection over union) or iof (intersection over foreground).

返回 Return the ious betweens boxes. If `aligned` is `False`, the shape of ious is (m, n) else (m, 1).

返回类型 torch.Tensor

示例

```
>>> bboxes1 = torch.FloatTensor([
>>>     [0, 0, 10, 10],
>>>     [10, 10, 20, 20],
>>>     [32, 32, 38, 42],
>>> ])
>>> bboxes2 = torch.FloatTensor([
>>>     [0, 0, 10, 20],
>>>     [0, 10, 10, 19],
>>>     [10, 10, 20, 20],
>>> ])
>>> bbox_overlaps(bboxes1, bboxes2)
tensor([[0.5000, 0.0000, 0.0000],
```

(下页继续)

(续上页)

```
        [0.0000, 0.0000, 1.0000],
        [0.0000, 0.0000, 0.0000]])
```

示例

```
>>> empty = torch.FloatTensor([])
>>> nonempty = torch.FloatTensor([
>>>     [0, 0, 10, 9],
>>> ])
>>> assert tuple(bbox_overlaps(empty, nonempty).shape) == (0, 1)
>>> assert tuple(bbox_overlaps(nonempty, empty).shape) == (1, 0)
>>> assert tuple(bbox_overlaps(empty, empty).shape) == (0, 0)
```

22.59 mmcv.ops.border_align

mmcv.ops.**border_align**()

22.60 mmcv.ops.box_iou_rotated

mmcv.ops.**box_iou_rotated**(*bboxes1: torch.Tensor*, *bboxes2: torch.Tensor*, *mode: str = 'iou'*, *aligned: bool = False*, *clockwise: bool = True*) → torch.Tensor

Return intersection-over-union (Jaccard index) of boxes.

Both sets of boxes are expected to be in (x_center, y_center, width, height, angle) format.

If `aligned` is `False`, then calculate the ious between each bbox of bboxes1 and bboxes2, otherwise the ious between each aligned pair of bboxes1 and bboxes2.

注解： The operator assumes:

1) The positive direction along x axis is left -> right.

2) The positive direction along y axis is top -> down.

3) The w border is in parallel with x axis when angle = 0.

However, there are 2 opposite definitions of the positive angular direction, clockwise (CW) and counter-clockwise (CCW). MMCV supports both definitions and uses CW by default.

Please set `clockwise=False` if you are using the CCW definition.

The coordinate system when `clockwise` is `True` (default)

```
0-------------------> x (0 rad)
|  A-------------B
|  |             |
|  |     box     h
|  |   angle=0   |
|  D------w------C
v
y (pi/2 rad)
```

In such coordination system the rotation matrix is

$$\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

The coordinates of the corner point A can be calculated as:

$$\begin{aligned} P_A = \begin{pmatrix} x_A \\ y_A \end{pmatrix} &= \begin{pmatrix} x_{center} \\ y_{center} \end{pmatrix} + \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} -0.5w \\ -0.5h \end{pmatrix} \\ &= \begin{pmatrix} x_{center} - 0.5w\cos\alpha + 0.5h\sin\alpha \\ y_{center} - 0.5w\sin\alpha - 0.5h\cos\alpha \end{pmatrix} \end{aligned}$$

The coordinate system when `clockwise` is `False`

```
0-------------------> x (0 rad)
|  A-------------B
|  |             |
|  |     box     h
|  |   angle=0   |
|  D------w------C
v
y (-pi/2 rad)
```

In such coordination system the rotation matrix is

$$\begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$$

The coordinates of the corner point A can be calculated as:

$$\begin{aligned} P_A = \begin{pmatrix} x_A \\ y_A \end{pmatrix} &= \begin{pmatrix} x_{center} \\ y_{center} \end{pmatrix} + \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} -0.5w \\ -0.5h \end{pmatrix} \\ &= \begin{pmatrix} x_{center} - 0.5w\cos\alpha - 0.5h\sin\alpha \\ y_{center} + 0.5w\sin\alpha - 0.5h\cos\alpha \end{pmatrix} \end{aligned}$$

参数

- **boxes1** (*torch.Tensor*) –rotated bboxes 1. It has shape (N, 5), indicating (x, y, w, h, theta) for each row. Note that theta is in radian.
- **boxes2** (*torch.Tensor*) –rotated bboxes 2. It has shape (M, 5), indicating (x, y, w, h, theta) for each row. Note that theta is in radian.
- **mode** (*str*) – "iou" (intersection over union) or iof (intersection over foreground).
- **clockwise** (*bool*) –flag indicating whether the positive angular orientation is clockwise. default True. *New in version 1.4.3.*

返回 Return the ious betweens boxes. If `aligned` is `False`, the shape of ious is (N, M) else (N,).

返回类型 torch.Tensor

22.61 mmcv.ops.boxes_iou3d

mmcv.ops.**boxes_iou3d**(*boxes_a: torch.Tensor*, *boxes_b: torch.Tensor*) → torch.Tensor

Calculate boxes 3D IoU.

参数

- **boxes_a** (*torch.Tensor*) –Input boxes a with shape (M, 7).
- **boxes_b** (*torch.Tensor*) –Input boxes b with shape (N, 7).

返回 3D IoU result with shape (M, N).

返回类型 torch.Tensor

22.62 mmcv.ops.boxes_iou_bev

mmcv.ops.**boxes_iou_bev**(*boxes_a: torch.Tensor*, *boxes_b: torch.Tensor*) → torch.Tensor

Calculate boxes IoU in the Bird's Eye View.

参数

- **boxes_a** (*torch.Tensor*) –Input boxes a with shape (M, 5) ([x1, y1, x2, y2, ry]).
- **boxes_b** (*torch.Tensor*) –Input boxes b with shape (N, 5) ([x1, y1, x2, y2, ry]).

返回 IoU result with shape (M, N).

返回类型 torch.Tensor

22.63 mmcv.ops.boxes_overlap_bev

mmcv.ops.**boxes_overlap_bev**(*boxes_a: torch.Tensor*, *boxes_b: torch.Tensor*) → torch.Tensor

Calculate boxes BEV overlap.

参数

- **boxes_a** (*torch.Tensor*) –Input boxes a with shape (M, 7).
- **boxes_b** (*torch.Tensor*) –Input boxes b with shape (N, 7).

返回 BEV overlap result with shape (M, N).

返回类型 torch.Tensor

22.64 mmcv.ops.carafe

mmcv.ops.**carafe**()

22.65 mmcv.ops.carafe_naive

mmcv.ops.**carafe_naive**()

22.66 mmcv.ops.chamfer_distance

mmcv.ops.**chamfer_distance**()

22.67 mmcv.ops.contour_expand

mmcv.ops.**contour_expand**(*kernel_mask: Union[numpy.array, torch.Tensor]*, *internal_kernel_label: Union[numpy.array, torch.Tensor]*, *min_kernel_area: int*, *kernel_num: int*) → list

Expand kernel contours so that foreground pixels are assigned into instances.

参数

- **kernel_mask** (*np.array or torch.Tensor*) –The instance kernel mask with size hxw.
- **internal_kernel_label** (*np.array or torch.Tensor*) –The instance internal kernel label with size hxw.
- **min_kernel_area** (*int*) –The minimum kernel area.

- **kernel_num** (*int*) –The instance kernel number.

返回 The instance index map with size hxw.

返回类型 list

22.68 mmcv.ops.convex_giou

mmcv.ops.**convex_giou**(*pointsets: torch.Tensor*, *polygons: torch.Tensor*) → Tuple[torch.Tensor, torch.Tensor]

Return generalized intersection-over-union (Jaccard index) between point sets and polygons.

参数

- **pointsets** (*torch.Tensor*) –It has shape (N, 18), indicating (x1, y1, x2, y2, ···, x9, y9) for each row.
- **polygons** (*torch.Tensor*) –It has shape (N, 8), indicating (x1, y1, x2, y2, x3, y3, x4, y4) for each row.

返回 The first element is the gious between point sets and polygons with the shape (N,). The second element is the gradient of point sets with the shape (N, 18).

返回类型 tuple[torch.Tensor, torch.Tensor]

22.69 mmcv.ops.convex_iou

mmcv.ops.**convex_iou**(*pointsets: torch.Tensor*, *polygons: torch.Tensor*) → torch.Tensor

Return intersection-over-union (Jaccard index) between point sets and polygons.

参数

- **pointsets** (*torch.Tensor*) –It has shape (N, 18), indicating (x1, y1, x2, y2, ···, x9, y9) for each row.
- **polygons** (*torch.Tensor*) –It has shape (K, 8), indicating (x1, y1, x2, y2, x3, y3, x4, y4) for each row.

返回 Return the ious between point sets and polygons with the shape (N, K).

返回类型 torch.Tensor

22.70 mmcv.ops.deform_conv2d

mmcv.ops.**deform_conv2d**()

22.71 mmcv.ops.deform_roi_pool

mmcv.ops.**deform_roi_pool**()

22.72 mmcv.ops.diff_iou_rotated_2d

mmcv.ops.**diff_iou_rotated_2d**(*box1: torch.Tensor*, *box2: torch.Tensor*) → torch.Tensor

Calculate differentiable iou of rotated 2d boxes.

参数

- **box1** (*Tensor*) –(B, N, 5) First box.
- **box2** (*Tensor*) –(B, N, 5) Second box.

返回 (B, N) IoU.

返回类型 Tensor

22.73 mmcv.ops.diff_iou_rotated_3d

mmcv.ops.**diff_iou_rotated_3d**(*box3d1: torch.Tensor*, *box3d2: torch.Tensor*) → torch.Tensor

Calculate differentiable iou of rotated 3d boxes.

参数

- **box3d1** (*Tensor*) –(B, N, 3+3+1) First box (x,y,z,w,h,l,alpha).
- **box3d2** (*Tensor*) –(B, N, 3+3+1) Second box (x,y,z,w,h,l,alpha).

返回 (B, N) IoU.

返回类型 Tensor

22.74 mmcv.ops.dynamic_scatter

mmcv.ops.**dynamic_scatter**()

22.75 mmcv.ops.furthest_point_sample

mmcv.ops.**furthest_point_sample**()

22.76 mmcv.ops.furthest_point_sample_with_dist

mmcv.ops.**furthest_point_sample_with_dist**()

22.77 mmcv.ops.fused_bias_leakyrelu

mmcv.ops.**fused_bias_leakyrelu**(*input: torch.Tensor*, *bias: torch.nn.parameter.Parameter*, *negative_slope: float = 0.2*, *scale: float = 1.4142135623730951*) → torch.Tensor

Fused bias leaky ReLU function.

This function is introduced in the StyleGAN2: Analyzing and Improving the Image Quality of StyleGAN

The bias term comes from the convolution operation. In addition, to keep the variance of the feature map or gradients unchanged, they also adopt a scale similarly with Kaiming initialization. However, since the $1 + alpha^2$ is too small, we can just ignore it. Therefore, the final scale is just $\sqrt{2}$. Of course, you may change it with your own scale.

参数

- **input** (*torch.Tensor*) –Input feature map.
- **bias** (*nn.Parameter*) –The bias from convolution operation.
- **negative_slope** (*float, optional*) –Same as nn.LeakyRelu. Defaults to 0.2.
- **scale** (*float, optional*) –A scalar to adjust the variance of the feature map. Defaults to 2**0.5.

返回 Feature map after non-linear activation.

返回类型 torch.Tensor

22.78 mmcv.ops.gather_points

mmcv.ops.**gather_points**()

22.79 mmcv.ops.grouping_operation

mmcv.ops.**grouping_operation**()

22.80 mmcv.ops.knn

mmcv.ops.**knn**()

22.81 mmcv.ops.masked_conv2d

mmcv.ops.**masked_conv2d**()

22.82 mmcv.ops.min_area_polygons

mmcv.ops.**min_area_polygons**(*pointsets: torch.Tensor*) → torch.Tensor

Find the smallest polygons that surrounds all points in the point sets.

参数 **pointsets** (*Tensor*) –point sets with shape (N, 18).

返回 Return the smallest polygons with shape (N, 8).

返回类型 torch.Tensor

22.83 mmcv.ops.modulated_deform_conv2d

mmcv.ops.**modulated_deform_conv2d**()

22.84 mmcv.ops.nms

mmcv.ops.**nms**(*boxes: Union[torch.Tensor, numpy.ndarray]*, *scores: Union[torch.Tensor, numpy.ndarray]*, *iou_threshold: float*, *offset: int = 0*, *score_threshold: float = 0*, *max_num: int = - 1*) → Tuple[Union[torch.Tensor, numpy.ndarray], Union[torch.Tensor, numpy.ndarray]]

Dispatch to either CPU or GPU NMS implementations.

The input can be either torch tensor or numpy array. GPU NMS will be used if the input is gpu tensor, otherwise CPU NMS will be used. The returned type will always be the same as inputs.

参数

- **boxes** (*torch.Tensor or np.ndarray*) –boxes in shape (N, 4).
- **scores** (*torch.Tensor or np.ndarray*) –scores in shape (N,).
- **iou_threshold** (*float*) –IoU threshold for NMS.
- **offset** (*int, 0 or 1*) –boxes' width or height is (x2 - x1 + offset).
- **score_threshold** (*float*) –score threshold for NMS.
- **max_num** (*int*) –maximum number of boxes after NMS.

返回 kept dets (boxes and scores) and indice, which always have the same data type as the input.

返回类型 tuple

示例

```
>>> boxes = np.array([[49.1, 32.4, 51.0, 35.9],
>>>                   [49.3, 32.9, 51.0, 35.3],
>>>                   [49.2, 31.8, 51.0, 35.4],
>>>                   [35.1, 11.5, 39.1, 15.7],
>>>                   [35.6, 11.8, 39.3, 14.2],
>>>                   [35.3, 11.5, 39.9, 14.5],
>>>                   [35.2, 11.7, 39.7, 15.7]], dtype=np.float32)
>>> scores = np.array([0.9, 0.9, 0.5, 0.5, 0.5, 0.4, 0.3],                dtype=np.
↪float32)
>>> iou_threshold = 0.6
>>> dets, inds = nms(boxes, scores, iou_threshold)
>>> assert len(inds) == len(dets) == 3
```

22.85 mmcv.ops.nms3d

mmcv.ops.**nms3d**(*boxes: torch.Tensor*, *scores: torch.Tensor*, *iou_threshold: float*) → torch.Tensor

3D NMS function GPU implementation (for BEV boxes).

参数

- **boxes** (*torch.Tensor*) –Input boxes with the shape of (N, 7) ([x, y, z, dx, dy, dz, heading]).
- **scores** (*torch.Tensor*) –Scores of boxes with the shape of (N).
- **iou_threshold** (*float*) –Overlap threshold of NMS.

返回 Indexes after NMS.

返回类型 torch.Tensor

22.86 mmcv.ops.nms3d_normal

mmcv.ops.**nms3d_normal**(*boxes: torch.Tensor*, *scores: torch.Tensor*, *iou_threshold: float*) → torch.Tensor

Normal 3D NMS function GPU implementation. The overlap of two boxes for IoU calculation is defined as the exact overlapping area of the two boxes WITH their yaw angle set to 0.

参数

- **boxes** (*torch.Tensor*) –Input boxes with shape (N, 7). ([x, y, z, dx, dy, dz, heading]).
- **scores** (*torch.Tensor*) –Scores of predicted boxes with shape (N).
- **iou_threshold** (*float*) –Overlap threshold of NMS.

返回 Remaining indices with scores in descending order.

返回类型 torch.Tensor

22.87 mmcv.ops.nms_bev

mmcv.ops.**nms_bev**(*boxes: torch.Tensor*, *scores: torch.Tensor*, *thresh: float*, *pre_max_size: Optional[int] = None*, *post_max_size: Optional[int] = None*) → torch.Tensor

NMS function GPU implementation (for BEV boxes).

The overlap of two boxes for IoU calculation is defined as the exact overlapping area of the two boxes. In this function, one can also set `pre_max_size` and `post_max_size`.

参数

- **boxes** (*torch.Tensor*) –Input boxes with the shape of (N, 5) ([x1, y1, x2, y2, ry]).

- **scores** (*torch.Tensor*) –Scores of boxes with the shape of (N,).
- **thresh** (*float*) –Overlap threshold of NMS.
- **pre_max_size** (*int, optional*) –Max size of boxes before NMS. Default: None.
- **post_max_size** (*int, optional*) –Max size of boxes after NMS. Default: None.

返回 Indexes after NMS.

返回类型 torch.Tensor

22.88 mmcv.ops.nms_match

mmcv.ops.**nms_match**(*dets: Union[torch.Tensor, numpy.ndarray]*, *iou_threshold: float*) → List[Union[numpy.ndarray, torch.Tensor]]

Matched dets into different groups by NMS.

NMS match is Similar to NMS but when a bbox is suppressed, nms match will record the indice of suppressed bbox and form a group with the indice of kept bbox. In each group, indice is sorted as score order.

参数

- **dets** (*torch.Tensor | np.ndarray*) –Det boxes with scores, shape (N, 5).
- **iou_threshold** (*float*) –IoU thresh for NMS.

返回 The outer list corresponds different matched group, the inner Tensor corresponds the indices for a group in score order.

返回类型 list[torch.Tensor | np.ndarray]

22.89 mmcv.ops.nms_normal_bev

mmcv.ops.**nms_normal_bev**(*boxes: torch.Tensor*, *scores: torch.Tensor*, *thresh: float*) → torch.Tensor

Normal NMS function GPU implementation (for BEV boxes).

The overlap of two boxes for IoU calculation is defined as the exact overlapping area of the two boxes WITH their yaw angle set to 0.

参数

- **boxes** (*torch.Tensor*) –Input boxes with shape (N, 5) ([x1, y1, x2, y2, ry]).
- **scores** (*torch.Tensor*) –Scores of predicted boxes with shape (N,).
- **thresh** (*float*) –Overlap threshold of NMS.

返回 Remaining indices with scores in descending order.

返回类型 torch.Tensor

22.90 mmcv.ops.nms_rotated

mmcv.ops.**nms_rotated**(*dets: torch.Tensor*, *scores: torch.Tensor*, *iou_threshold: float*, *labels: Optional[torch.Tensor] = None*, *clockwise: bool = True*) → Tuple[torch.Tensor, torch.Tensor]

Performs non-maximum suppression (NMS) on the rotated boxes according to their intersection-over-union (IoU).

Rotated NMS iteratively removes lower scoring rotated boxes which have an IoU greater than iou_threshold with another (higher scoring) rotated box.

参数

- **dets** (*torch.Tensor*) –Rotated boxes in shape (N, 5). They are expected to be in (x_ctr, y_ctr, width, height, angle_radian) format.
- **scores** (*torch.Tensor*) –scores in shape (N,).
- **iou_threshold** (*float*) –IoU thresh for NMS.
- **labels** (*torch.Tensor, optional*) –boxes' label in shape (N,).
- **clockwise** (*bool*) –flag indicating whether the positive angular orientation is clockwise. default True. *New in version 1.4.3.*

返回 kept dets(boxes and scores) and indice, which is always the same data type as the input.

返回类型 tuple

22.91 mmcv.ops.pixel_group

mmcv.ops.**pixel_group**(*score: Union[numpy.ndarray, torch.Tensor]*, *mask: Union[numpy.ndarray, torch.Tensor]*, *embedding: Union[numpy.ndarray, torch.Tensor]*, *kernel_label: Union[numpy.ndarray, torch.Tensor]*, *kernel_contour: Union[numpy.ndarray, torch.Tensor]*, *kernel_region_num: int*, *distance_threshold: float*) → List[List[float]]

Group pixels into text instances, which is widely used text detection methods.

参数

- **score** (*np.array or torch.Tensor*) –The foreground score with size hxw.
- **mask** (*np.array or Tensor*) –The foreground mask with size hxw.
- **embedding** (*np.array or torch.Tensor*) –The embedding with size hxwxc to distinguish instances.
- **kernel_label** (*np.array or torch.Tensor*) –The instance kernel index with size hxw.

- **kernel_contour** (*np.array or torch.Tensor*) –The kernel contour with size hxw.
- **kernel_region_num** (*int*) –The instance kernel region number.
- **distance_threshold** (*float*) –The embedding distance threshold between kernel and pixel in one instance.

返回 The instance coordinates and attributes list. Each element consists of averaged confidence, pixel number, and coordinates (x_i, y_i for all pixels) in order.

返回类型 list[list[float]]

22.92 mmcv.ops.point_sample

mmcv.ops.**point_sample**(*input: torch.Tensor*, *points: torch.Tensor*, *align_corners: bool = False*, ***kwargs*) → torch.Tensor

A wrapper around `grid_sample()` to support 3D point_coords tensors Unlike `torch.nn.functional.grid_sample()` it assumes point_coords to lie inside `[0, 1] x [0, 1]` square.

参数

- **input** (*torch.Tensor*) –Feature map, shape (N, C, H, W).
- **points** (*torch.Tensor*) –Image based absolute point coordinates (normalized), range [0, 1] x [0, 1], shape (N, P, 2) or (N, Hgrid, Wgrid, 2).
- **align_corners** (*bool, optional*) –Whether align_corners. Default: False

返回 Features of *point* on *input*, shape (N, C, P) or (N, C, Hgrid, Wgrid).

返回类型 torch.Tensor

22.93 mmcv.ops.points_in_boxes_all

mmcv.ops.**points_in_boxes_all**(*points: torch.Tensor*, *boxes: torch.Tensor*) → torch.Tensor

Find all boxes in which each point is (CUDA).

参数

- **points** (*torch.Tensor*) –[B, M, 3], [x, y, z] in LiDAR/DEPTH coordinate
- **boxes** (*torch.Tensor*) –[B, T, 7], num_valid_boxes <= T, [x, y, z, x_size, y_size, z_size, rz], (x, y, z) is the bottom center.

返回 Return the box indices of points with the shape of (B, M, T). Default background = 0.

返回类型 torch.Tensor

22.94 mmcv.ops.points_in_boxes_cpu

mmcv.ops.**points_in_boxes_cpu**(*points: torch.Tensor*, *boxes: torch.Tensor*) → torch.Tensor

Find all boxes in which each point is (CPU). The CPU version of *points_in_boxes_all()*.

参数

- **points** (*torch.Tensor*) –[B, M, 3], [x, y, z] in LiDAR/DEPTH coordinate
- **boxes** (*torch.Tensor*) –[B, T, 7], num_valid_boxes <= T, [x, y, z, x_size, y_size, z_size, rz], (x, y, z) is the bottom center.

返回 Return the box indices of points with the shape of (B, M, T). Default background = 0.

返回类型 torch.Tensor

22.95 mmcv.ops.points_in_boxes_part

mmcv.ops.**points_in_boxes_part**(*points: torch.Tensor*, *boxes: torch.Tensor*) → torch.Tensor

Find the box in which each point is (CUDA).

参数

- **points** (*torch.Tensor*) –[B, M, 3], [x, y, z] in LiDAR/DEPTH coordinate.
- **boxes** (*torch.Tensor*) –[B, T, 7], num_valid_boxes <= T, [x, y, z, x_size, y_size, z_size, rz] in LiDAR/DEPTH coordinate, (x, y, z) is the bottom center.

返回 Return the box indices of points with the shape of (B, M). Default background = -1.

返回类型 torch.Tensor

22.96 mmcv.ops.points_in_polygons

mmcv.ops.**points_in_polygons**(*points: torch.Tensor*, *polygons: torch.Tensor*) → torch.Tensor

Judging whether points are inside polygons, which is used in the ATSS assignment for the rotated boxes.

It should be noted that when the point is just at the polygon boundary, the judgment will be inaccurate, but the effect on assignment is limited.

参数

- **points** (*torch.Tensor*) –It has shape (B, 2), indicating (x, y). M means the number of predicted points.
- **polygons** (*torch.Tensor*) –It has shape (M, 8), indicating (x1, y1, x2, y2, x3, y3, x4, y4). M means the number of ground truth polygons.

返回 Return the result with the shape of (B, M), 1 indicates that the point is inside the polygon, 0 indicates that the point is outside the polygon.

返回类型 torch.Tensor

22.97 mmcv.ops.prroi_pool

mmcv.ops.**prroi_pool**()

22.98 mmcv.ops.rel_roi_point_to_rel_img_point

mmcv.ops.**rel_roi_point_to_rel_img_point**(*rois: torch.Tensor*, *rel_roi_points: torch.Tensor*, *img: Union[tuple, torch.Tensor]*, *spatial_scale: float = 1.0*) → torch.Tensor

Convert roi based relative point coordinates to image based absolute point coordinates.

参数

- **rois** (*torch.Tensor*) –RoIs or BBoxes, shape (N, 4) or (N, 5)
- **rel_roi_points** (*torch.Tensor*) –Point coordinates inside RoI, relative to RoI, location, range (0, 1), shape (N, P, 2)
- **img** (*tuple or torch.Tensor*) –(height, width) of image or feature map.
- **spatial_scale** (*float, optional*) –Scale points by this factor. Default: 1.

返回 Image based relative point coordinates for sampling, shape (N, P, 2).

返回类型 torch.Tensor

22.99 mmcv.ops.riroi_align_rotated

mmcv.ops.**riroi_align_rotated**()

22.100 mmcv.ops.roi_align

mmcv.ops.**roi_align**()

22.101 mmcv.ops.roi_align_rotated

mmcv.ops.**roi_align_rotated**()

22.102 mmcv.ops.roi_pool

mmcv.ops.**roi_pool**()

22.103 mmcv.ops.rotated_feature_align

mmcv.ops.**rotated_feature_align**(*features: torch.Tensor*, *best_rbboxes: torch.Tensor*, *spatial_scale: float = 0.125*, *points: int = 1*) → torch.Tensor

22.104 mmcv.ops.scatter_nd

mmcv.ops.**scatter_nd**(*indices: torch.Tensor*, *updates: torch.Tensor*, *shape: torch.Tensor*) → torch.Tensor

pytorch edition of tensorflow scatter_nd.

this function don't contain except handle code. so use this carefully when indice repeats, don't support repeat add which is supported in tensorflow.

22.105 mmcv.ops.sigmoid_focal_loss

mmcv.ops.**sigmoid_focal_loss**()

22.106 mmcv.ops.soft_nms

mmcv.ops.**soft_nms**(*boxes: Union[torch.Tensor, numpy.ndarray]*, *scores: Union[torch.Tensor, numpy.ndarray]*, *iou_threshold: float = 0.3*, *sigma: float = 0.5*, *min_score: float = 0.001*, *method: str = 'linear'*, *offset: int = 0*) → Tuple[Union[torch.Tensor, numpy.ndarray], Union[torch.Tensor, numpy.ndarray]]

Dispatch to only CPU Soft NMS implementations.

The input can be either a torch tensor or numpy array. The returned type will always be the same as inputs.

参数

- **boxes** (*torch.Tensor or np.ndarray*) –boxes in shape (N, 4).

- **scores** (*torch.Tensor or np.ndarray*) –scores in shape (N,).
- **iou_threshold** (*float*) –IoU threshold for NMS.
- **sigma** (*float*) –hyperparameter for gaussian method
- **min_score** (*float*) –score filter threshold
- **method** (*str*) –either ‘linear’ or ‘gaussian’
- **offset** (*int, 0 or 1*) –boxes’ width or height is (x2 - x1 + offset).

返回 kept dets (boxes and scores) and indice, which always have the same data type as the input.

返回类型 tuple

示例

```
>>> boxes = np.array([[4., 3., 5., 3.],
>>>                   [4., 3., 5., 4.],
>>>                   [3., 1., 3., 1.],
>>>                   [3., 1., 3., 1.],
>>>                   [3., 1., 3., 1.],
>>>                   [3., 1., 3., 1.]], dtype=np.float32)
>>> scores = np.array([0.9, 0.9, 0.5, 0.5, 0.4, 0.0], dtype=np.float32)
>>> iou_threshold = 0.6
>>> dets, inds = soft_nms(boxes, scores, iou_threshold, sigma=0.5)
>>> assert len(inds) == len(dets) == 5
```

22.107 mmcv.ops.softmax_focal_loss

mmcv.ops.**softmax_focal_loss**()

22.108 mmcv.ops.three_interpolate

mmcv.ops.**three_interpolate**()

22.109 mmcv.ops.three_nn

mmcv.ops.**three_nn**()

22.110 mmcv.ops.tin_shift

mmcv.ops.**tin_shift**()

22.111 mmcv.ops.upfirdn2d

mmcv.ops.**upfirdn2d**(*input: torch.Tensor*, *filter: torch.Tensor*, *up: int = 1*, *down: int = 1*, *padding: Union[int, List[int]] = 0*, *flip_filter: bool = False*, *gain: Union[float, int] = 1*, *use_custom_op: bool = True*)

Pad, upsample, filter, and downsample a batch of 2D images.

Performs the following sequence of operations for each channel:

1. Upsample the image by inserting N-1 zeros after each pixel (*up*).

2. Pad the image with the specified number of zeros on each side (*padding*). Negative padding corresponds to cropping the image.

3. Convolve the image with the specified 2D FIR filter (*f*), shrinking it so that the footprint of all output pixels lies within the input image.

4. Downsample the image by keeping every Nth pixel (*down*).

This sequence of operations bears close resemblance to scipy.signal.upfirdn().

The fused op is considerably more efficient than performing the same calculation using standard PyTorch ops. It supports gradients of arbitrary order.

参数

- **input** (*torch.Tensor*) –Float32/float64/float16 input tensor of the shape *[batch_size, num_channels, in_height, in_width]*.
- **filter** (*torch.Tensor*) –Float32 FIR filter of the shape *[filter_height, filter_width]* (non-separable), *[filter_taps]* (separable), or *None* (identity).
- **up** (*int*) –Integer upsampling factor. Can be a single int or a list/tuple *[x, y]*. Defaults to 1.
- **down** (*int*) –Integer downsampling factor. Can be a single int or a list/tuple *[x, y]*. Defaults to 1.

- **padding** (*int | tuple[int]*) –Padding with respect to the upsampled image. Can be a single number or a list/tuple *[x, y]* or *[x_before, x_after, y_before, y_after]*. Defaults to 0.
- **flip_filter** (*bool*) –False = convolution, True = correlation. Defaults to False.
- **gain** (*int*) –Overall scaling factor for signal magnitude. Defaults to 1.
- **use_custom_op** (*bool*) –Whether to use customized op. Defaults to True.

返回 Tensor of the shape *[batch_size, num_channels, out_height, out_width]*

22.112 mmcv.ops.voxelization

mmcv.ops.**voxelization**()

CHAPTER 23

mmcv.transforms

BaseTransform	Base class for all transformations.
TestTimeAug	Test-time augmentation transform.

23.1 BaseTransform

class mmcv.transforms.**BaseTransform**

Base class for all transformations.

abstract transform(*results: Dict*) → Optional[Union[Dict, Tuple[List, List]]]

The transform function. All subclass of BaseTransform should override this method.

This function takes the result dict as the input, and can add new items to the dict or modify existing items in the dict. And the result dict will be returned in the end, which allows to concate multiple transforms into a pipeline.

参数 **results** (*dict*) –The result dict.

返回 The result dict.

返回类型 dict

23.2 TestTimeAug

class mmcv.transforms.**TestTimeAug**(*transforms: list*)

Test-time augmentation transform.

An example configuration is as followed:

```
dict(type='TestTimeAug',
     transforms=[
        [dict(type='Resize', scale=(1333, 400), keep_ratio=True),
         dict(type='Resize', scale=(1333, 800), keep_ratio=True)],
        [dict(type='RandomFlip', prob=1.),
         dict(type='RandomFlip', prob=0.)],
        [dict(type='PackDetInputs',
              meta_keys=('img_id', 'img_path', 'ori_shape',
                         'img_shape', 'scale_factor', 'flip',
                         'flip_direction'))]])
```

`results` will be transformed using all transforms defined in `transforms` arguments.

For the above configuration, there are four combinations of resize and flip:

- Resize to (1333, 400) + no flip
- Resize to (1333, 400) + flip
- Resize to (1333, 800) + no flip
- resize to (1333, 800) + flip

After that, results are wrapped into lists of the same length as below:

```
dict(
    inputs=[...],
    data_samples=[...]
)
```

The length of `inputs` and `data_samples` are both 4.

Required Keys:

- Depending on the requirements of the `transforms` parameter.

Modified Keys:

- All output keys of each transform.

参数 transforms (*list[list[dict]]*) –Transforms to be applied to data sampled from dataset. `transforms` is a list of list, and each list element usually represents a series of trans-

forms with the same type and different arguments. Data will be processed by each list elements sequentially. See more information in *transform()*.

transform(*results: dict*) → dict

Apply all transforms defined in `transforms` to the results.

As the example given in *TestTimeAug*, `transforms` consists of 2 `Resize`, 2 `RandomFlip` and 1 `PackDetInputs`. The data sampled from dataset will be processed as follows:

1. Data will be processed by 2 `Resize` and return a list of 2 results.
2. Each result in list will be further passed to 2 `RandomFlip`, and aggregates into a list of 4 results.
3. Each result will be processed by `PackDetInputs`, and return a list of dict.
4. Aggregates the same fields of results, and finally returns a dict. Each value of the dict represents 4 transformed results.

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

返回 The augmented data, where each value is wrapped into a list.

返回类型 dict

23.3 Loading

LoadAnnotations	Load and process the `instances` and `seg_map` annotation provided by dataset.
LoadImageFromFile	Load an image from file.

23.3.1 LoadAnnotations

class mmcv.transforms.**LoadAnnotations**(*with_bbox: bool = True*, *with_label: bool = True*, *with_seg: bool = False*, *with_keypoints: bool = False*, *imdecode_backend: str = 'cv2'*, *file_client_args: Optional[dict] = None*, *, *backend_args: Optional[dict] = None*)

Load and process the `instances` and `seg_map` annotation provided by dataset.

The annotation format is as the following:

```
{
    'instances':
    [
        {
        # List of 4 numbers representing the bounding box of the
```

(下页继续)

(续上页)

```
        # instance, in (x1, y1, x2, y2) order.
        'bbox': [x1, y1, x2, y2],

        # Label of image classification.
        'bbox_label': 1,

        # Used in key point detection.
        # Can only load the format of [x1, y1, v1,…, xn, yn, vn]. v[i]
        # means the visibility of this keypoint. n must be equal to the
        # number of keypoint categories.
        'keypoints': [x1, y1, v1, ..., xn, yn, vn]
        }
    ]
    # Filename of semantic or panoptic segmentation ground truth file.
    'seg_map_path': 'a/b/c'
}
```

After this module, the annotation has been changed to the format below:

```
{
    # In (x1, y1, x2, y2) order, float type. N is the number of bboxes
    # in np.float32
    'gt_bboxes': np.ndarray(N, 4)
     # In np.int64 type.
    'gt_bboxes_labels': np.ndarray(N, )
     # In uint8 type.
    'gt_seg_map': np.ndarray (H, W)
     # with (x, y, v) order, in np.float32 type.
    'gt_keypoints': np.ndarray(N, NK, 3)
}
```

Required Keys:

- instances
  - bbox (optional)
  - bbox_label
  - keypoints (optional)
- seg_map_path (optional)

Added Keys:

- gt_bboxes (np.float32)
- gt_bboxes_labels (np.int64)

- gt_seg_map (np.uint8)
- gt_keypoints (np.float32)

参数

- **with_bbox** (*bool*) –Whether to parse and load the bbox annotation. Defaults to True.
- **with_label** (*bool*) –Whether to parse and load the label annotation. Defaults to True.
- **with_seg** (*bool*) –Whether to parse and load the semantic segmentation annotation. Defaults to False.
- **with_keypoints** (*bool*) –Whether to parse and load the keypoints annotation. Defaults to False.
- **imdecode_backend** (*str*) –The image decoding backend type. The backend argument for `mmcv.imfrombytes()`. See `mmcv.imfrombytes()` for details. Defaults to 'cv2'.
- **file_client_args** (*dict, optional*) –Arguments to instantiate a FileClient. See `mmengine.fileio.FileClient` for details. Defaults to None. It will be deprecated in future. Please use `backend_args` instead. Deprecated in version 2.0.0rc4.
- **backend_args** (*dict, optional*) –Instantiates the corresponding file backend. It may contain *backend* key to specify the file backend. If it contains, the file backend corresponding to this value will be used and initialized with the remaining values, otherwise the corresponding file backend will be selected based on the prefix of the file path. Defaults to None. New in version 2.0.0rc4.

transform(*results: dict*) → dict

Function to load multiple types annotations.

参数 **results** (*dict*) –Result dict from `mmengine.dataset.BaseDataset`.

返回 The dict contains loaded bounding box, label and semantic segmentation and keypoints annotations.

返回类型 dict

23.3.2 LoadImageFromFile

class mmcv.transforms.**LoadImageFromFile**(*to_float32: bool = False*, *color_type: str = 'color'*, *imdecode_backend: str = 'cv2'*, *file_client_args: Optional[dict] = None*, *ignore_empty: bool = False*, ***, *backend_args: Optional[dict] = None*)

Load an image from file.

Required Keys:

- img_path

Modified Keys:

- img
- img_shape
- ori_shape

参数

- **to_float32** (*bool*) –Whether to convert the loaded image to a float32 numpy array. If set to False, the loaded image is an uint8 array. Defaults to False.
- **color_type** (*str*) –The flag argument for `mmcv.imfrombytes()`. Defaults to 'color'.
- **imdecode_backend** (*str*) –The image decoding backend type. The backend argument for `mmcv.imfrombytes()`. See `mmcv.imfrombytes()` for details. Defaults to 'cv2'.
- **file_client_args** (*dict, optional*) –Arguments to instantiate a FileClient. See `mmengine.fileio.FileClient` for details. Defaults to None. It will be deprecated in future. Please use `backend_args` instead. Deprecated in version 2.0.0rc4.
- **ignore_empty** (*bool*) –Whether to allow loading empty image or file path not existent. Defaults to False.
- **backend_args** (*dict, optional*) –Instantiates the corresponding file backend. It may contain *backend* key to specify the file backend. If it contains, the file backend corresponding to this value will be used and initialized with the remaining values, otherwise the corresponding file backend will be selected based on the prefix of the file path. Defaults to None. New in version 2.0.0rc4.

transform(*results: dict*) → Optional[dict]

Functions to load image.

参数 **results** (*dict*) –Result dict from `mmengine.dataset.BaseDataset`.

返回 The dict contains loaded image and meta information.

返回类型 dict

23.4 Processing

CenterCrop	Crop the center of the image, segmentation masks, bounding boxes and key points.
MultiScaleFlipAug	Test-time augmentation with multiple scales and flipping.
Normalize	Normalize the image.
Pad	Pad the image & segmentation map.
RandomChoiceResize	Resize images & bbox & mask from a list of multiple scales.
RandomFlip	Flip the image & bbox & keypoints & segmentation map.
RandomGrayscale	Randomly convert image to grayscale with a probability.
RandomResize	Random resize images & bbox & keypoints.
Resize	Resize images & bbox & seg & keypoints.
ToTensor	Convert some results to `torch.Tensor` by given keys.
ImageToTensor	Convert image to `torch.Tensor` by given keys.

23.4.1 CenterCrop

class mmcv.transforms.**CenterCrop**(*crop_size: Union[int, Tuple[int, int]]*, *auto_pad: bool = False*, *pad_cfg: dict = {'type': 'Pad'}*, *clip_object_border: bool = True*)

Crop the center of the image, segmentation masks, bounding boxes and key points. If the crop area exceeds the original image and `auto_pad` is True, the original image will be padded before cropping.

Required Keys:

- img
- gt_seg_map (optional)
- gt_bboxes (optional)
- gt_keypoints (optional)

Modified Keys:

- img
- img_shape
- gt_seg_map (optional)
- gt_bboxes (optional)
- gt_keypoints (optional)

Added Key:

- pad_shape

参数

- **crop_size** (*Union[int, Tuple[int, int]]*) –Expected size after cropping with the format of (w, h). If set to an integer, then cropping width and height are equal to this integer.
- **auto_pad** (*bool*) –Whether to pad the image if it's smaller than the `crop_size`. Defaults to False.
- **pad_cfg** (*dict*) –Base config for padding. Refer to `mmcv.Pad` for detail. Defaults to `dict(type='Pad')`.
- **clip_object_border** (*bool*) –Whether to clip the objects outside the border of the image. In some dataset like MOT17, the gt bboxes are allowed to cross the border of images. Therefore, we don't need to clip the gt bboxes in these cases. Defaults to True.

transform (*results: dict*) → dict

Apply center crop on results.

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

返回 Results with CenterCropped image and semantic segmentation map.

返回类型 dict

23.4.2 MultiScaleFlipAug

class mmcv.transforms.**MultiScaleFlipAug**(*transforms: List[dict]*, *scales: Optional[Union[Tuple, List[Tuple]]] = None*, *scale_factor: Optional[Union[float, List[float]]] = None*, *allow_flip: bool = False*, *flip_direction: Union[str, List[str]] = 'horizontal'*, *resize_cfg: dict = {'keep_ratio': True, 'type': 'Resize'}*, *flip_cfg: dict = {'type': 'RandomFlip'}*)

Test-time augmentation with multiple scales and flipping.

An example configuration is as followed:

```
dict(
    type='MultiScaleFlipAug',
    scales=[(1333, 400), (1333, 800)],
    flip=True,
    transforms=[
        dict(type='Normalize', **img_norm_cfg),
        dict(type='Pad', size_divisor=1),
        dict(type='ImageToTensor', keys=['img']),
```

(下页继续)

(续上页)

```
        dict(type='Collect', keys=['img'])
    ])
```

`results` will be resized using all the sizes in `scales`. If `flip` is True, then flipped results will also be added into output list.

For the above configuration, there are four combinations of resize and flip:

- Resize to (1333, 400) + no flip
- Resize to (1333, 400) + flip
- Resize to (1333, 800) + no flip
- resize to (1333, 800) + flip

The four results are then transformed with `transforms` argument. After that, results are wrapped into lists of the same length as below:

```
dict(
    inputs=[...],
    data_samples=[...]
)
```

Where the length of `inputs` and `data_samples` are both 4.

Required Keys:

- Depending on the requirements of the `transforms` parameter.

Modified Keys:

- All output keys of each transform.

参数

- **transforms** (*list[dict]*) –Transforms to be applied to each resized and flipped data.
- **scales** (*tuple | list[tuple] | None*) –Images scales for resizing.
- **scale_factor** (*float or tuple[float]*) –Scale factors for resizing. Defaults to None.
- **allow_flip** (*bool*) –Whether apply flip augmentation. Defaults to False.
- **flip_direction** (*str | list[str]*) –Flip augmentation directions, options are "horizontal", "vertical" and "diagonal". If flip_direction is a list, multiple flip augmentations will be applied. It has no effect when flip == False. Defaults to "horizontal".
- **resize_cfg** (*dict*) –Base config for resizing. Defaults to `dict(type='Resize', keep_ratio=True)`.

- **flip_cfg** (*dict*) –Base config for flipping. Defaults to `dict(type='RandomFlip')`.

transform(*results: dict*) → Dict

Apply test time augment transforms on results.

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

返回 The augmented data, where each value is wrapped into a list.

返回类型 dict

23.4.3 Normalize

class mmcv.transforms.**Normalize**(*mean: Sequence[Union[int, float]]*, *std: Sequence[Union[int, float]]*, *to_rgb: bool = True*)

Normalize the image.

Required Keys:

- img

Modified Keys:

- img

Added Keys:

- img_norm_cfg
  - mean
  - std
  - to_rgb

参数

- **mean** (*sequence*) –Mean values of 3 channels.
- **std** (*sequence*) –Std values of 3 channels.
- **to_rgb** (*bool*) –Whether to convert the image from BGR to RGB before normlizing the image. If `to_rgb=True`, the order of mean and std should be RGB. If `to_rgb=False`, the order of mean and std should be the same order of the image. Defaults to True.

transform(*results: dict*) → dict

Function to normalize images.

参数 **results** (*dict*) –Result dict from loading pipeline.

返回 Normalized results, key ‘img_norm_cfg’ key is added in to result dict.

返回类型 dict

23.4.4 Pad

class mmcv.transforms.**Pad**(*size: Optional[Tuple[int, int]] = None*, *size_divisor: Optional[int] = None*, *pad_to_square: bool = False*, *pad_val: Union[int, float, dict] = {'img': 0, 'seg': 255}*, *padding_mode: str = 'constant'*)

Pad the image & segmentation map.

There are three padding modes: (1) pad to a fixed size and (2) pad to the minimum size that is divisible by some number. and (3)pad to square. Also, pad to square and pad to the minimum size can be used as the same time.

Required Keys:

- img
- gt_bboxes (optional)
- gt_seg_map (optional)

Modified Keys:

- img
- gt_seg_map
- img_shape

Added Keys:

- pad_shape
- pad_fixed_size
- pad_size_divisor

参数

- **size** (*tuple, optional*) –Fixed padding size. Expected padding shape (w, h). Defaults to None.
- **size_divisor** (*int, optional*) –The divisor of padded size. Defaults to None.
- **pad_to_square** (*bool*) –Whether to pad the image into a square. Currently only used for YOLOX. Defaults to False.
- **pad_val** (*Number | dict[str, Number], optional*) –Padding value for if the pad_mode is "constant". If it is a single number, the value to pad the image is the number and to pad the semantic segmentation map is 255. If it is a dict, it should have the following keys:
  - img: The value to pad the image.

- seg: The value to pad the semantic segmentation map.

  Defaults to dict(img=0, seg=255).

- **padding_mode** (*str*) –Type of padding. Should be: constant, edge, reflect or symmetric. Defaults to ‘constant’.
  - constant: pads with a constant value, this value is specified with pad_val.
  - edge: pads with the last value at the edge of the image.
  - reflect: pads with reflection of image without repeating the last value on the edge. For example, padding [1, 2, 3, 4] with 2 elements on both sides in reflect mode will result in [3, 2, 1, 2, 3, 4, 3, 2].
  - symmetric: pads with reflection of image repeating the last value on the edge. For example, padding [1, 2, 3, 4] with 2 elements on both sides in symmetric mode will result in [2, 1, 1, 2, 3, 4, 4, 3]

transform(*results: dict*) → dict

Call function to pad images, masks, semantic segmentation maps.

参数 **results** (*dict*) –Result dict from loading pipeline.

返回 Updated result dict.

返回类型 dict

23.4.5 RandomChoiceResize

class mmcv.transforms.**RandomChoiceResize**(*scales: Sequence[Union[int, Tuple]]*, *resize_type: str = 'Resize'*, ***resize_kwargs*)

Resize images & bbox & mask from a list of multiple scales.

This transform resizes the input image to some scale. Bboxes and masks are then resized with the same scale factor. Resize scale will be randomly selected from `scales`.

How to choose the target scale to resize the image will follow the rules below:

- if *scale* is a list of tuple, the target scale is sampled from the list uniformally.
- if *scale* is a tuple, the target scale will be set to the tuple.

Required Keys:

- img
- gt_bboxes (optional)
- gt_seg_map (optional)
- gt_keypoints (optional)

Modified Keys:

- img
- img_shape
- gt_bboxes (optional)
- gt_seg_map (optional)
- gt_keypoints (optional)

Added Keys:

- scale
- scale_factor
- scale_idx
- keep_ratio

参数

- **scales** (*Union[list, Tuple]*) –Images scales for resizing.
- **resize_type** (*str*) –The type of resize class to use. Defaults to "Resize".
- ****resize_kwargs** –Other keyword arguments for the `resize_type`.

注解: By defaults, the `resize_type` is "Resize", if it's not overwritten by your registry, it indicates the `mmcv.Resize`. And therefore, `resize_kwargs` accepts any keyword arguments of it, like `keep_ratio`, `interpolation` and so on.

If you want to use your custom resize class, the class should accept `scale` argument and have `scale` attribution which determines the resize shape.

---

transform(*results: dict*) → dict

Apply resize transforms on results from a list of scales.

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

返回 Resized results, 'img', 'gt_bboxes', 'gt_seg_map', 'gt_keypoints', 'scale', 'scale_factor', 'img_shape', and 'keep_ratio' keys are updated in result dict.

返回类型 dict

23.4.6 RandomFlip

class mmcv.transforms.**RandomFlip**(*prob: Optional[Union[float, Iterable[float]]] = None*, *direction: Union[str, Sequence[Optional[str]]] = 'horizontal'*, *swap_seg_labels: Optional[Sequence] = None*)

Flip the image & bbox & keypoints & segmentation map. Added or Updated keys: flip, flip_direction, img, gt_bboxes, gt_seg_map, and gt_keypoints. There are 3 flip modes:

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

Required Keys:

- img
- gt_bboxes (optional)
- gt_seg_map (optional)
- gt_keypoints (optional)

Modified Keys:

- img
- gt_bboxes (optional)
- gt_seg_map (optional)
- gt_keypoints (optional)

Added Keys:

- flip
- flip_direction
- swap_seg_labels (optional)

参数

- **prob** (*float | list[float], optional*) –The flipping probability. Defaults to None.
- **direction** (*str | list[str]*) –The flipping direction. Options If input is a list, the length must equal `prob`. Each element in `prob` indicates the flip probability of corresponding direction. Defaults to 'horizontal'.
- **swap_seg_labels** (*list, optional*) –The label pair need to be swapped for ground truth, like 'left arm' and 'right arm' need to be swapped after horizontal flipping. For example, `[(1, 5)]`, where 1/5 is the label of the left/right arm. Defaults to None.

transform (*results: dict*) → dict

Transform function to flip images, bounding boxes, semantic segmentation map and keypoints.

参数 **results** (*dict*) –Result dict from loading pipeline.

返回 Flipped results, 'img', 'gt_bboxes', 'gt_seg_map', 'gt_keypoints', 'flip', and 'flip_direction' keys are updated in result dict.

返回类型 dict

23.4.7 RandomGrayscale

class mmcv.transforms.**RandomGrayscale**(*prob: float = 0.1*, *keep_channels: bool = False*, *channel_weights: Sequence[float] = (1.0, 1.0, 1.0)*, *color_format: str = 'bgr'*)

Randomly convert image to grayscale with a probability.

Required Key:

- img

Modified Key:

- img

Added Keys:

- grayscale
- grayscale_weights

参数

- **prob** (*float*) –Probability that image should be converted to grayscale. Defaults to 0.1.
- **keep_channels** (*bool*) –Whether keep channel number the same as input. Defaults to False.

- **channel_weights** (*tuple*) –The grayscale weights of each channel, and the weights will be normalized. For example, (1, 2, 1) will be normalized as (0.25, 0.5, 0.25). Defaults to (1., 1., 1.).
- **color_format** (*str*) –Color format set to be any of ‘bgr’, ‘rgb’, ‘hsv’. Note: ‘hsv’ image will be transformed into ‘bgr’ format no matter whether it is grayscaled. Defaults to ‘bgr’.

transform (*results: dict*) → dict

Apply random grayscale on results.

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

返回 Results with grayscale image.

返回类型 dict

23.4.8 RandomResize

class mmcv.transforms.**RandomResize**(*scale: Union[Tuple[int, int], Sequence[Tuple[int, int]]]*, *ratio_range: Optional[Tuple[float, float]] = None*, *resize_type: str = 'Resize'*, ***resize_kwargs*)

Random resize images & bbox & keypoints.

How to choose the target scale to resize the image will follow the rules below:

- if `scale` is a sequence of tuple

$$target\_scale[0] \sim Uniform([scale[0][0], scale[1][0]])$$

$$target\_scale[1] \sim Uniform([scale[0][1], scale[1][1]])$$

Following the resize order of weight and height in cv2, `scale[i][0]` is for width, and `scale[i][1]` is for height.

- if `scale` is a tuple

$$target\_scale[0] \sim Uniform([ratio\_range[0], ratio\_range[1]]) * scale[0]$$

$$target\_scale[0] \sim Uniform([ratio\_range[0], ratio\_range[1]]) * scale[1]$$

Following the resize order of weight and height in cv2, `ratio_range[0]` is for width, and `ratio_range[1]` is for height.

- if `keep_ratio` is True, the minimum value of `target_scale` will be used to set the shorter side and the maximum value will be used to set the longer side.
- if `keep_ratio` is False, the value of `target_scale` will be used to reisze the width and height accordingly.

Required Keys:

- img
- gt_bboxes
- gt_seg_map
- gt_keypoints

Modified Keys:

- img
- gt_bboxes
- gt_seg_map
- gt_keypoints
- img_shape

Added Keys:

- scale
- scale_factor
- keep_ratio

参数

- **scale** (*tuple or Sequence[tuple]*) –Images scales for resizing. Defaults to None.
- **ratio_range** (*tuple[float], optional*) –(min_ratio, max_ratio). Defaults to None.
- **resize_type** (*str*) –The type of resize class to use. Defaults to "Resize".
- ****resize_kwargs** –Other keyword arguments for the `resize_type`.

注解: By defaults, the `resize_type` is "Resize", if it's not overwritten by your registry, it indicates the `mmcv.Resize`. And therefore, `resize_kwargs` accepts any keyword arguments of it, like `keep_ratio`, `interpolation` and so on.

If you want to use your custom resize class, the class should accept `scale` argument and have `scale` attribution which determines the resize shape.

---

transform (*results: dict*) → dict

Transform function to resize images, bounding boxes, semantic segmentation map.

参数 **results** (*dict*) –Result dict from loading pipeline.

返回 Resized results, `img`, `gt_bboxes`, `gt_semantic_seg`, `gt_keypoints`, `scale`, `scale_factor`, `img_shape`, and `keep_ratio` keys are updated in result dict.

返回类型 dict

23.4.9 Resize

class mmcv.transforms.**Resize**(*scale: Optional[Union[int, Tuple[int, int]]] = None*, *scale_factor: Optional[Union[float, Tuple[float, float]]] = None*, *keep_ratio: bool = False*, *clip_object_border: bool = True*, *backend: str = 'cv2'*, *interpolation='bilinear'*)

Resize images & bbox & seg & keypoints.

This transform resizes the input image according to `scale` or `scale_factor`. Bboxes, seg map and keypoints are then resized with the same scale factor. if `scale` and `scale_factor` are both set, it will use `scale` to resize.

Required Keys:

- img
- gt_bboxes (optional)
- gt_seg_map (optional)
- gt_keypoints (optional)

Modified Keys:

- img
- gt_bboxes
- gt_seg_map
- gt_keypoints
- img_shape

Added Keys:

- scale
- scale_factor
- keep_ratio

参数

- **scale** (*int or tuple*) –Images scales for resizing. Defaults to None
- **scale_factor** (*float or tuple[float]*) –Scale factors for resizing. Defaults to None.
- **keep_ratio** (*bool*) –Whether to keep the aspect ratio when resizing the image. Defaults to False.

- **clip_object_border** (*bool*) –Whether to clip the objects outside the border of the image. In some dataset like MOT17, the gt bboxes are allowed to cross the border of images. Therefore, we don't need to clip the gt bboxes in these cases. Defaults to True.
- **backend** (*str*) –Image resize backend, choices are 'cv2' and 'pillow'. These two backends generates slightly different results. Defaults to 'cv2'.
- **interpolation** (*str*) –Interpolation method, accepted values are "nearest", "bilinear", "bicubic", "area", "lanczos" for 'cv2' backend, "nearest", "bilinear" for 'pillow' backend. Defaults to 'bilinear'.

transform(*results: dict*) → dict

Transform function to resize images, bounding boxes, semantic segmentation map and keypoints.

参数 **results** (*dict*) –Result dict from loading pipeline.

返回 Resized results, 'img', 'gt_bboxes', 'gt_seg_map', 'gt_keypoints', 'scale', 'scale_factor', 'img_shape', and 'keep_ratio' keys are updated in result dict.

返回类型 dict

23.4.10 ToTensor

class mmcv.transforms.**ToTensor**(*keys: Sequence[str]*)

Convert some results to `torch.Tensor` by given keys.

Required keys:

- all these keys in *keys*

Modified Keys:

- all these keys in *keys*

参数 **keys** (*Sequence[str]*) –Keys that need to be converted to Tensor.

transform(*results: dict*) → dict

Transform function to convert data to *torch.Tensor*.

参数 **results** (*dict*) –Result dict from loading pipeline.

返回 *keys* in results will be updated.

返回类型 dict

23.4.11 ImageToTensor

class mmcv.transforms.**ImageToTensor**(*keys: dict*)

Convert image to `torch.Tensor` by given keys.

The dimension order of input image is (H, W, C). The pipeline will convert it to (C, H, W). If only 2 dimension (H, W) is given, the output would be (1, H, W).

Required keys:

- all these keys in *keys*

Modified Keys:

- all these keys in *keys*

参数 **keys** (*Sequence[str]*) –Key of images to be converted to Tensor.

transform(*results: dict*) → dict

Transform function to convert image in results to `torch.Tensor` and transpose the channel order. :param results: Result dict contains the image data to convert. :type results: dict

返回 The result dict contains the image converted to :obj:`torch.Tensor` and transposed to (C, H, W) order.

返回类型 dict

23.5 Wrapper

Compose	Compose multiple transforms sequentially.
KeyMapper	A transform wrapper to map and reorganize the input/output of the wrapped transforms (or sub-pipeline).
RandomApply	Apply transforms randomly with a given probability.
RandomChoice	Process data with a randomly chosen transform from given candidates.
TransformBroadcaster	A transform wrapper to apply the wrapped transforms to multiple data items.

23.5.1 Compose

class mmcv.transforms.**Compose**(*transforms: Union[Dict, Callable[[Dict], Dict], Sequence[Union[Dict, Callable[[Dict], Dict]]]]*)

Compose multiple transforms sequentially.

参数 **transforms** (*list[dict | callable]*) –Sequence of transform object or config dict to be composed.

实际案例

```
>>> pipeline = [
>>>     dict(type='Compose',
>>>         transforms=[
>>>             dict(type='LoadImageFromFile'),
>>>             dict(type='Normalize')
>>>         ]
>>>     )
>>> ]
```

transform(*results: Dict*) → Optional[Dict]

Call function to apply transforms sequentially.

参数 **results** (*dict*) –A result dict contains the results to transform.

返回 Transformed results.

返回类型 dict or None

23.5.2 KeyMapper

class mmcv.transforms.**KeyMapper**(*transforms: Optional[Union[Dict, Callable[[Dict], Dict], List[Union[Dict, Callable[[Dict], Dict]]]]] = None*, *mapping: Optional[Dict] = None*, *remapping: Optional[Dict] = None*, *auto_remap: Optional[bool] = None*, *allow_nonexist_keys: bool = False*)

A transform wrapper to map and reorganize the input/output of the wrapped transforms (or sub-pipeline).

参数

- **transforms** (*list[dict | callable], optional*) –Sequence of transform object or config dict to be wrapped.
- **mapping** (*dict*) –A dict that defines the input key mapping. The keys corresponds to the inner key (i.e., kwargs of the `transform` method), and should be string type. The values

corresponds to the outer keys (i.e., the keys of the data/results), and should have a type of string, list or dict. None means not applying input mapping. Default: None.

- **remapping** (*dict*) –A dict that defines the output key mapping. The keys and values have the same meanings and rules as in the `mapping`. Default: None.
- **auto_remap** (*bool, optional*) –If True, an inverse of the mapping will be used as the remapping. If auto_remap is not given, it will be automatically set True if ‘remapping’ is not given, and vice versa. Default: None.
- **allow_nonexist_keys** (*bool*) –If False, the outer keys in the mapping must exist in the input data, or an exception will be raised. Default: False.

实际案例

```
>>> # Example 1: KeyMapper 'gt_img' to 'img'
>>> pipeline = [
>>>     # Use KeyMapper to convert outer (original) field name
>>>     # 'gt_img' to inner (used by inner transforms) filed name
>>>     # 'img'
>>>     dict(type='KeyMapper',
>>>         mapping={'img': 'gt_img'},
>>>         # auto_remap=True means output key mapping is the revert of
>>>         # the input key mapping, e.g. inner 'img' will be mapped
>>>         # back to outer 'gt_img'
>>>         auto_remap=True,
>>>         transforms=[
>>>             # In all transforms' implementation just use 'img'
>>>             # as a standard field name
>>>             dict(type='Crop', crop_size=(384, 384)),
>>>             dict(type='Normalize'),
>>>         ])
>>> ]
```

```
>>> # Example 2: Collect and structure multiple items
>>> pipeline = [
>>>     # The inner field 'imgs' will be a dict with keys 'img_src'
>>>     # and 'img_tar', whose values are outer fields 'img1' and
>>>     # 'img2' respectively.
>>>     dict(type='KeyMapper',
>>>         dict(
>>>             type='KeyMapper',
>>>             mapping=dict(
>>>                 imgs=dict(
```

(下页继续)

(续上页)

```
>>>                     img_src='img1',
>>>                     img_tar='img2')),
>>>         transforms=...)
>>> ]
```

```
>>> # Example 3: Manually set ignored keys by "..."
>>> pipeline = [
>>>     ...
>>>     dict(type='KeyMapper',
>>>         mapping={
>>>             # map outer key "gt_img" to inner key "img"
>>>             'img': 'gt_img',
>>>             # ignore outer key "mask"
>>>             'mask': ...,
>>>         },
>>>         transforms=[
>>>             dict(type='RandomFlip'),
>>>         ])
>>>     ...
>>> ]
```

transform(*results: Dict*) → Dict

Apply mapping, wrapped transforms and remapping.

23.5.3 RandomApply

class mmcv.transforms.**RandomApply**(*transforms: Union[Dict, Callable[[Dict], Dict], List[Union[Dict, Callable[[Dict], Dict]]]]*, *prob: float = 0.5*)

Apply transforms randomly with a given probability.

参数

- **transforms** (*list[dict | callable]*) –The transform or transform list to randomly apply.
- **prob** (*float*) –The probability to apply transforms. Default: 0.5

实际案例

```
>>> # config
>>> pipeline = [
>>>     dict(type='RandomApply',
>>>         transforms=[dict(type='HorizontalFlip')],
>>>         prob=0.3)
>>> ]
```

transform(*results: Dict*) → Optional[Dict]

Randomly apply the transform.

23.5.4 RandomChoice

class mmcv.transforms.**RandomChoice**(*transforms: List[Union[Dict, Callable[[Dict], Dict], List[Union[Dict, Callable[[Dict], Dict]]]]]*, *prob: Optional[List[float]] = None*)

Process data with a randomly chosen transform from given candidates.

参数

- **transforms** (*list[list]*) –A list of transform candidates, each is a sequence of transforms.
- **prob** (*list[float], optional*) –The probabilities associated with each pipeline. The length should be equal to the pipeline number and the sum should be 1. If not given, a uniform distribution will be assumed.

实际案例

```
>>> # config
>>> pipeline = [
>>>     dict(type='RandomChoice',
>>>         transforms=[
>>>             [dict(type='RandomHorizontalFlip')],  # subpipeline 1
>>>             [dict(type='RandomRotate')],  # subpipeline 2
>>>         ]
>>>     )
>>> ]
```

transform(*results: Dict*) → Optional[Dict]

Randomly choose a transform to apply.

23.5.5 TransformBroadcaster

class mmcv.transforms.**TransformBroadcaster**(*transforms: List[Union[Dict, Callable[[Dict], Dict]]]*, *mapping: Optional[Dict] = None*, *remapping: Optional[Dict] = None*, *auto_remap: Optional[bool] = None*, *allow_nonexist_keys: bool = False*, *share_random_params: bool = False*)

A transform wrapper to apply the wrapped transforms to multiple data items. For example, apply Resize to multiple images.

参数

- **transforms** (*list[dict | callable]*) –Sequence of transform object or config dict to be wrapped.
- **mapping** (*dict*) –A dict that defines the input key mapping. Note that to apply the transforms to multiple data items, the outer keys of the target items should be remapped as a list with the standard inner key (The key required by the wrapped transform). See the following example and the document of `mmcv.transforms.wrappers.KeyMapper` for details.
- **remapping** (*dict*) –A dict that defines the output key mapping. The keys and values have the same meanings and rules as in the `mapping`. Default: None.
- **auto_remap** (*bool, optional*) –If True, an inverse of the mapping will be used as the remapping. If auto_remap is not given, it will be automatically set True if 'remapping' is not given, and vice versa. Default: None.
- **allow_nonexist_keys** (*bool*) –If False, the outer keys in the mapping must exist in the input data, or an exception will be raised. Default: False.
- **share_random_params** (*bool*) –If True, the random transform (e.g., RandomFlip) will be conducted in a deterministic way and have the same behavior on all data items. For example, to randomly flip either both input image and ground-truth image, or none. Default: False.

注解: To apply the transforms to each elements of a list or tuple, instead of separating data items, you can map the outer key of the target sequence to the standard inner key. See example 2. example.

实际案例

```
>>> # Example 1: Broadcast to enumerated keys, each contains a single
>>> # data element
>>> pipeline = [
>>>     dict(type='LoadImageFromFile', key='lq'),  # low-quality img
>>>     dict(type='LoadImageFromFile', key='gt'),  # ground-truth img
>>>     # TransformBroadcaster maps multiple outer fields to standard
>>>     # the inner field and process them with wrapped transforms
>>>     # respectively
>>>     dict(type='TransformBroadcaster',
>>>         # case 1: from multiple outer fields
>>>         mapping={'img': ['lq', 'gt']},
>>>         auto_remap=True,
>>>         # share_random_param=True means using identical random
>>>         # parameters in every processing
>>>         share_random_param=True,
>>>         transforms=[
>>>             dict(type='Crop', crop_size=(384, 384)),
>>>             dict(type='Normalize'),
>>>         ])
>>> ]
```

```
>>> # Example 2: Broadcast to keys that contains data sequences
>>> pipeline = [
>>>     dict(type='LoadImageFromFile', key='lq'),  # low-quality img
>>>     dict(type='LoadImageFromFile', key='gt'),  # ground-truth img
>>>     # TransformBroadcaster maps multiple outer fields to standard
>>>     # the inner field and process them with wrapped transforms
>>>     # respectively
>>>     dict(type='TransformBroadcaster',
>>>         # case 2: from one outer field that contains multiple
>>>         # data elements (e.g. a list)
>>>         # mapping={'img': 'images'},
>>>         auto_remap=True,
>>>         share_random_param=True,
>>>         transforms=[
>>>             dict(type='Crop', crop_size=(384, 384)),
>>>             dict(type='Normalize'),
>>>         ])
>>> ]
```

```
>>> Example 3: Set ignored keys in broadcasting
>>> pipeline = [
```

(下页继续)

(续上页)

```
>>>         dict(type='TransformBroadcaster',
>>>             # Broadcast the wrapped transforms to multiple images
>>>             # 'lq' and 'gt, but only update 'img_shape' once
>>>             mapping={
>>>                 'img': ['lq', 'gt'],
>>>                 'img_shape': ['img_shape', ...],
>>>              },
>>>             auto_remap=True,
>>>             share_random_params=True,
>>>             transforms=[
>>>                 # `RandomCrop` will modify the field "img",
>>>                 # and optionally update "img_shape" if it exists
>>>                 dict(type='RandomCrop'),
>>>             ])
>>>     ]
```

scatter_sequence(*data: Dict*) → List[Dict]

Scatter the broadcasting targets to a list of inputs of the wrapped transforms.

transform(*results: Dict*)

Broadcast wrapped transforms to multiple targets.

CHAPTER 24

mmcv.arraymisc

quantize	Quantize an array of (-inf, inf) to [0, levels-1].
dequantize	Dequantize an array.

24.1 mmcv.arraymisc.quantize

mmcv.arraymisc.**quantize**(*arr: numpy.ndarray, min_val: Union[int, float], max_val: Union[int, float], levels: int, dtype=<class 'numpy.int64'>*) → tuple

Quantize an array of (-inf, inf) to [0, levels-1].

参数

- **arr** (*ndarray*) –Input array.
- **min_val** (*int or float*) –Minimum value to be clipped.
- **max_val** (*int or float*) –Maximum value to be clipped.
- **levels** (*int*) –Quantization levels.
- **dtype** (*np.type*) –The type of the quantized array.

返回 Quantized array.

返回类型 tuple

24.2 mmcv.arraymisc.dequantize

mmcv.arraymisc.**dequantize**(*arr: numpy.ndarray, min_val: Union[int, float], max_val: Union[int, float], levels: int, dtype=<class 'numpy.float64'>*) → tuple

Dequantize an array.

参数

- **arr** (*ndarray*) –Input array.
- **min_val** (*int or float*) –Minimum value to be clipped.
- **max_val** (*int or float*) –Maximum value to be clipped.
- **levels** (*int*) –Quantization levels.
- **dtype** (*np.type*) –The type of the dequantized array.

返回 Dequantized array.

返回类型 tuple

CHAPTER 25

mmcv.utils

IS_CUDA_AVAILABLE	bool(x) -> bool
IS_MLU_AVAILABLE	bool(x) -> bool
IS_MPS_AVAILABLE	bool(x) -> bool
collect_env	Collect the information of the running environments.
jit	
skip_no_elena	

25.1 mmcv.utils.IS_CUDA_AVAILABLE

mmcv.utils.**IS_CUDA_AVAILABLE = False**

bool(x) -> bool

Returns True when the argument x is true, False otherwise. The builtins True and False are the only two instances of the class bool. The class bool is a subclass of the class int, and cannot be subclassed.

25.2 mmcv.utils.IS_MLU_AVAILABLE

mmcv.utils.**IS_MLU_AVAILABLE = False**

bool(x) -> bool

Returns True when the argument x is true, False otherwise. The builtins True and False are the only two instances of the class bool. The class bool is a subclass of the class int, and cannot be subclassed.

25.3 mmcv.utils.IS_MPS_AVAILABLE

mmcv.utils.**IS_MPS_AVAILABLE = False**

bool(x) -> bool

Returns True when the argument x is true, False otherwise. The builtins True and False are the only two instances of the class bool. The class bool is a subclass of the class int, and cannot be subclassed.

25.4 mmcv.utils.collect_env

mmcv.utils.**collect_env**()

Collect the information of the running environments.

返回

The environment information. The following fields are contained.

- sys.platform: The variable of `sys.platform`.
- Python: Python version.
- CUDA available: Bool, indicating if CUDA is available.
- GPU devices: Device type of each GPU.
- CUDA_HOME (optional): The env var `CUDA_HOME`.
- NVCC (optional): NVCC version.
- GCC: GCC version, "n/a" if GCC is not installed.
- MSVC: Microsoft Virtual C++ Compiler version, Windows only.
- PyTorch: PyTorch version.
- PyTorch compiling details: The output of `torch.__config__.show()`.
- TorchVision (optional): TorchVision version.
- OpenCV: OpenCV version.

- MMEngine: MMEngine version.
- MMCV: MMCV version.
- MMCV Compiler: The GCC version for compiling MMCV ops.
- MMCV CUDA Compiler: The CUDA version for compiling MMCV ops.

返回类型 dict

25.5 mmcv.utils.jit

mmcv.utils.**jit**(*func=None*, *check_input=None*, *full_shape=True*, *derivate=False*, *coderize=False*, *optimize=False*)

25.6 mmcv.utils.skip_no_elena

mmcv.utils.**skip_no_elena**(*func*)

CHAPTER 26

Indices and tables

- genindex
- search

索引

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y