MMOCR

发布 *1.0.0rc0*

OpenMMLab

2022 年 09 月 18 日

新手入门

您可以在页面左下角切换中英文文档。

CHAPTER 1

概览

MMOCR 是一个基于 PyTorch 和 MMDetection 的开源工具箱，支持众多 OCR 相关的模型，涵盖了文本检测、文本识别以及关键信息提取等多个主要方向。它还支持了大多数流行的学术数据集，并提供了许多实用工具帮助用户对数据集和模型进行多方面的探索和调试，助力优质模型的产出和落地。它具有以下特点：

- **全流程，多模型**：支持了全流程的 OCR 任务，包括文本检测、文本识别及关键信息提取的各种最新模型。
- **模块化设计**：MMOCR 的模块化设计使用户可以按需定义及复用模型中的各个模块。
- **实用工具众多**：MMOCR 提供了全面的可视化工具、验证工具和性能评测工具，帮助用户对模型进行排错、调优或客观比较。
- **由 OpenMMLab 强力驱动**：与家族内的其它算法库一样，MMOCR 遵循着 OpenMMLab 严谨的开发准则和接口约定，极大地降低了用户切换各算法库时的学习成本。同时，MMOCR 也可以非常便捷地与家族内其他算法库跨库联动，从而满足用户跨领域研究和落地的需求。

随着 OpenMMLab 家族架构的整体升级，MMOCR 也相应地进行了大幅度的升级和修改。在这个大版本的更新中，MMOCR 中大量的冗余代码和重复实现被移除，多个关键方法的运行效率得到了提升，且整体框架设计上变得更为统一。考虑到该版本相较于 0.x 存在一些后向不兼容的修改，我们准备了一份详细的迁移指南，并在里面列出了新版本所作出的所有改动和迁移所需的步骤，力求帮助熟悉旧版框架的用户尽快完成升级。尽管这可能需要一定时间，但我们相信由 MMOCR 和 OpenMMLab 生态系统整体带来的新特性会让这一切变得尤为值得。

接下来，请根据实际需求选择你需要阅读的章节。我们推荐初学者通过【新手入门-快速运行】来熟悉 MMOCR 模型的调用，并从【用户指南】提供的案例中逐步掌握 MMOCR 的用法。中高级开发者则可以从【基础概念】中了解各个组件的背景、约定和推荐实现。同时，如果你在文档中未能找到需要的答案，欢迎通过 issue 进行反馈，也可以通过提 pull request 参与至 MMOCR 的社区建设中来。

CHAPTER 2

安装

2.1 环境依赖

- Linux | Windows | macOS
- Python 3.7
- PyTorch 1.6 或更高版本
- torchvision 0.7.0
- CUDA 10.1
- NCCL 2
- GCC 5.4.0 或更高版本

2.2 准备环境

注解： 如果你已经在本地安装了 PyTorch，请直接跳转到安装步骤。

第一步下载并安装 Miniconda.

第二步创建并激活一个 conda 环境：

```
conda create --name openmmlab python=3.8 -y
conda activate openmmlab
```

第三步依照官方指南，安装 PyTorch。

在 GPU 平台上：

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

在 CPU 平台上：

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

2.3 安装步骤

我们建议大多数用户采用我们的推荐方式安装 MMOCR。倘若你需要更灵活的安装过程，则可以参考自定义安装一节。

2.3.1 推荐步骤

第一步使用 MIM 安装 MMEngine and MMCV.

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

第二步将 MMDetection 以依赖库的形式安装。

```
pip install 'mmdet>=3.0.0rc0'
```

第三步安装 MMOCR.

情况 1: 若你需要直接运行 MMOCR 或在其基础上进行开发，则通过源码安装：

```
git clone https://github.com/open-mmlab/mmocr.git
cd mmocr
git checkout 1.x
pip install -r requirements.txt
pip install -v -e .
# "-v" 会让安装过程产生更详细的输出
# "-e" 会以可编辑的方式安装该代码库，你对该代码库所作的任何更改都会立即生效
```

情况 2：如果你将 MMOCR 作为一个外置依赖库使用，通过 pip 安装即可：

```
pip install 'mmocr>=1.0.0rc0'
```

第四步（可选）如果你需要使用与 albumentations 有关的变换，比如 ABINet 数据流水线中的 Albu，请使用以下命令安装依赖：

```
# 若 MMOCR 通过源码安装
pip install -r requirements/albu.txt
# 若 MMOCR 通过 pip 安装
pip install albumentations>=1.1.0 --no-binary qudida,albumentations
```

注解: 我们建议在安装 albumentations 之后检查当前环境，确保 opencv-python 和 opencv-python-headless 没有同时被安装，否则有可能会产生一些无法预知的错误。如果它们不巧同时存在于环境当中，请卸载 opencv-python-headless 以确保 MMOCR 的可视化工具可以正常运行。

查看 albumentations 的官方文档以获知详情。

2.3.2 检验

根据安装方式的不同，我们提供了验证安装正确性的方法。若 MMOCR 的安装无误，你在这一节完成后应当能看到以图片和文字形式表示的识别结果，示意如下：

```
# 识别结果
{'rec_texts': ['cbanke', 'docece', 'sroumats', 'chounsonse', 'doceca', 'c', '', 'sond
↪', 'abrandso', 'sretane', '1', 'tosl', 'roundi', 'slen', 'yet', 'ally', 's', 'sue',
↪'salle', 'v'], 'rec_scores': [...], 'det_polygons': [...], 'det_scores': tensor([...
↪])}
```

在 MMOCR 的目录运行以下命令：

```
python mmocr/ocr.py --det DB_r18 --recog CRNN demo/demo_text_ocr.jpg --show
```

也可以在 Python 解释器中运行以下代码：

```
from mmocr.utils.ocr import MMOCR
ocr = MMOCR(recog='CRNN', det='DB_r18')
ocr.readtext('demo_text_ocr.jpg', show=True)
```

2.4 自定义安装

2.4.1 CUDA 版本

安装 PyTorch 时，需要指定 CUDA 版本。如果您不清楚选择哪个，请遵循我们的建议：

- 对于 Ampere 架构的 NVIDIA GPU，例如 GeForce 30 series 以及 NVIDIA A100，CUDA 11 是必需的。
- 对于更早的 NVIDIA GPU，CUDA 11 是向前兼容的，但 CUDA 10.2 能够提供更好的兼容性，也更加轻量。

请确保你的 GPU 驱动版本满足最低的版本需求，参阅这张表。

注解： 如果按照我们的最佳实践进行安装，CUDA 运行时库就足够了，因为我们提供相关 CUDA 代码的预编译，你不需要进行本地编译。但如果你希望从源码进行 MMCV 的编译，或是进行其他 CUDA 算子的开发，那么就必须安装完整的 CUDA 工具链，参见 NVIDIA 官网，另外还需要确保该 CUDA 工具链的版本与 PyTorch 安装时的配置相匹配（如用 `conda install` 安装 PyTorch 时指定的 cudatoolkit 版本）。

2.4.2 不使用 MIM 安装 MMCV

MMCV 包含 C++ 和 CUDA 扩展，因此其对 PyTorch 的依赖比较复杂。MIM 会自动解析这些依赖，选择合适的 MMCV 预编译包，使安装更简单，但它并不是必需的。

要使用 pip 而不是 MIM 来安装 MMCV，请遵照 MMCV 安装指南。它需要你用指定 url 的形式手动指定对应的 PyTorch 和 CUDA 版本。

举个例子，如下命令将会安装基于 PyTorch 1.10.x 和 CUDA 11.3 编译的 mmcv-full。

```
pip install 'mmcv>=2.0.0rc1' -f https://download.openmmlab.com/mmcv/dist/cu113/torch1.
↪10/index.html
```

2.4.3 在 CPU 环境中安装

MMOCR 可以仅在 CPU 环境中安装，在 CPU 模式下，你可以完成训练（需要 MMCV 版本 >= 1.4.4）、测试和模型推理等所有操作。

在 CPU 模式下，MMCV 中的以下算子将不可用：

- Deformable Convolution
- Modulated Deformable Convolution
- ROI pooling
- SyncBatchNorm

如果你尝试使用用到了以上算子的模型进行训练、测试或推理，程序将会报错。以下为可能受到影响的模型列表：

2.4.4 通过 Docker 使用 MMOCR

我们提供了一个 Dockerfile 文件以建立 docker 镜像。

```
# build an image with PyTorch 1.6, CUDA 10.1
docker build -t mmocr docker/
```

使用以下命令运行。

```
docker run --gpus all --shm-size=8g -it -v {实际数据目录}:/mmocr/data mmocr
```

2.5 对 MMCV 和 MMDetection 的版本依赖

为了确保代码实现的正确性，MMOCR 每个版本都有可能改变对 MMCV 和 MMDetection 版本的依赖。请根据以下表格确保版本之间的相互匹配。

CHAPTER 3

快速运行

3.1 推理

如果想快速运行一个推理，请直接阅读安装文档的检验。对 MMOCR 中推理接口更为详细说明，可以在这里找到。

注解： 除了使用我们提供好的预训练模型，用户也可以在自己的数据集上训练流行模型。接下来我们以在迷你的 ICDAR 2015 数据集上训练 DBNet 为例，带大家熟悉 MMOCR 的基本功能。

接下来的部分都假设你使用的是*编辑方式安装 MMOCR 代码库*。

3.2 准备数据集

由于 OCR 任务的数据集种类多样，格式不一，不利于多数据集的切换和联合训练，因此 MMOCR 约定了一种统一的数据格式，并针对常用的 OCR 数据集都提供了对应的转换脚本和教程。通常，要在 MMOCR 中使用数据集，你只需要按照对应步骤运行指令即可。

注解： 但我们亦深知，效率就是生命——尤其对想要快速上手 MMOCR 的你来说。

在这里，我们准备了一个用于演示的精简版 ICDAR 2015 数据集。下载我们预先准备好的压缩包，解压到 mmocr 的 `data/det/` 目录下，就能得到我们准备好的图片和标注文件。

```
wget https://download.openmmlab.com/mmocr/data/icdar2015/mini_icdar2015.tar.gz
mkdir -p data/det/
tar xzvf mini_icdar2015.tar.gz -C data/det/
```

3.3 修改配置

准备好数据集后，我们接下来就需要通过修改配置的方式指定训练集的位置和训练参数。

在这个例子中，我们将会训练一个以 resnet18 作为骨干网络（backbone）的 DBNet。由于 MMOCR 已经有针对完整 ICDAR 2015 数据集的配置（configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py），我们只需要在它的基础上作出一点修改。

我们首先需要修改数据集的路径。在这个配置中，大部分关键的配置文件都在 _base_ 中被导入，如数据库的配置就来自 configs/_base_/det_datasets/icdar2015.py。打开该文件，把第一行 ic15_det_data_root 指向的路径替换：

```
ic15_det_data_root = 'data/det/mini_icdar2015'
```

另外，因为数据集尺寸缩小了，我们也要相应地减少训练的轮次到 400，缩短验证和储存权重的间隔到 10 轮，并放弃学习率衰减策略。直接把以下几行配置放入 configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py 即可生效：

```
# 每 10 个 epoch 储存一次权重
default_hooks = dict(checkpoint=dict(type='CheckpointHook', interval=10), )
# 设置最大 epoch 数为 400，每 10 个 epoch 运行一次验证
train_cfg = dict(type='EpochBasedTrainLoop', max_epochs=400, val_interval=10)
# 令学习率为常量，即不进行学习率衰减
param_scheduler = [dict(type='ConstantLR', factor=1.0),]
```

这里，我们通过配置的继承机制将基础配置中的相应参数直接进行了改写。原本的字段分布在 configs/_base_/schedules/schedule_sgd_1200e.py 和 configs/_base_/textdet_default_runtime.py 中，感兴趣的读者可以自行查看。

小技巧： 关于配置文件更加详尽的说明，请参考此处。

3.4 可视化数据集

在正式开始训练前，我们还可以可视化一下经过训练过程中数据变换（*transforms*）后的图像。方法也很简单，把我们需要可视化的配置传入 browse_dataset.py 脚本即可：

```
python tools/analysis_tools/browse_dataset.py configs/textdet/dbnet/dbnet_resnet18_
↪fpnc_1200e_icdar2015.py
```

数据变换后的图片和标签会在弹窗中逐张被展示出来。

注解： 有关该脚本更详细的指南，请参考此处.

小技巧： 除了满足好奇心之外，可视化还可以帮助我们在训练前检查可能影响到模型表现的部分，如配置文件、数据集及数据变换中的问题。

3.5 训练

万事俱备，只欠东风。运行以下命令启动训练：

```
python tools/train.py configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py
```

根据系统情况，MMOCR 会自动使用最佳的设备进行训练。如果有 GPU，则会默认在第一张卡启动单卡训练。当开始看到 loss 的输出，就说明你已经成功启动了训练。

```
2022/08/22 18:42:22 - mmengine - INFO - Epoch(train) [1][5/7]  lr: 7.0000e-03 ␣
↪memory: 7730  data_time: 0.4496  loss_prob: 14.6061  loss_thr: 2.2904  loss_db: 0.
↪9879  loss: 17.8843  time: 1.8666
2022/08/22 18:42:24 - mmengine - INFO - Exp name: dbnet_resnet18_fpnc_1200e_icdar2015
2022/08/22 18:42:28 - mmengine - INFO - Epoch(train) [2][5/7]  lr: 7.0000e-03 ␣
↪memory: 6695  data_time: 0.2052  loss_prob: 6.7840  loss_thr: 1.4114  loss_db: 0.
↪9855  loss: 9.1809  time: 0.7506
2022/08/22 18:42:29 - mmengine - INFO - Exp name: dbnet_resnet18_fpnc_1200e_icdar2015
2022/08/22 18:42:33 - mmengine - INFO - Epoch(train) [3][5/7]  lr: 7.0000e-03 ␣
↪memory: 6690  data_time: 0.2101  loss_prob: 3.0700  loss_thr: 1.1800  loss_db: 0.
↪9967  loss: 5.2468  time: 0.6244
2022/08/22 18:42:33 - mmengine - INFO - Exp name: dbnet_resnet18_fpnc_1200e_icdar2015
```

在不指定额外参数时，训练的权重默认会被保存到 work_dirs/dbnet_resnet18_fpnc_1200e_icdar2015/ 下面，而日志则会保存在 work_dirs/dbnet_resnet18_fpnc_1200e_icdar2015/开始训练的时间戳/里。接下来，我们只需要耐心等待模型训练完成即可。

小技巧： 若需要了解训练的高级用法，如 CPU 训练、多卡训练及集群训练等，请查阅训练与测试。

3.6 测试

经过数十分钟的等待，模型顺利完成了 400 epochs 的训练。我们通过控制台的输出，观察到 DBNet 在最后一个 epoch 的表现最好，hmean 达到了 60.86：

```
08/22 19:24:52 - mmengine - INFO - Epoch(val) [400][100/100]  icdar/precision: 0.7285␣
↪ icdar/recall: 0.5226  icdar/hmean: 0.6086
```

注解： 它或许还没被训练到最优状态，但对于一个演示而言已经足够了。

然而，这个数值只反映了 DBNet 在迷你 ICDAR 2015 数据集上的性能。要想更加客观地评判它的检测能力，我们还要看看它在分布外数据集上的表现。例如，tests/data/det_toy_dataset 就是一个很小的真实数据集，我们可以用它来验证一下 DBNet 的实际性能。

在测试前，我们同样需要对数据集的位置做一下修改。打开 configs/_base_/det_datasets/icdar2015.py，修改 ic15_det_test 的 data_root 为 tests/data/det_toy_dataset：

```
# ...
ic15_det_test = dict(
    type='OCRDataset',
    data_root='tests/data/det_toy_dataset',
    # ...
    )
```

修改完毕，运行命令启动测试。

```
python tools/test.py configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py␣
↪work_dirs/dbnet_resnet18_fpnc_1200e_icdar2015/epoch_400.pth
```

得到输出：

```
08/21 21:45:59 - mmengine - INFO - Epoch(test) [5/10]    memory: 8562
08/21 21:45:59 - mmengine - INFO - Epoch(test) [10/10]    eta: 0:00:00  time: 0.4893 ␣
↪data_time: 0.0191  memory: 283
08/21 21:45:59 - mmengine - INFO - Evaluating hmean-iou...
08/21 21:45:59 - mmengine - INFO - prediction score threshold: 0.30, recall: 0.6190,␣
↪precision: 0.4815, hmean: 0.5417
08/21 21:45:59 - mmengine - INFO - prediction score threshold: 0.40, recall: 0.6190,␣
↪precision: 0.5909, hmean: 0.6047
```

(下页继续)

(续上页)

```
08/21 21:45:59 - mmengine - INFO - prediction score threshold: 0.50, recall: 0.6190,␣
↪precision: 0.6842, hmean: 0.6500
08/21 21:45:59 - mmengine - INFO - prediction score threshold: 0.60, recall: 0.6190,␣
↪precision: 0.7222, hmean: 0.6667
08/21 21:45:59 - mmengine - INFO - prediction score threshold: 0.70, recall: 0.3810,␣
↪precision: 0.8889, hmean: 0.5333
08/21 21:45:59 - mmengine - INFO - prediction score threshold: 0.80, recall: 0.0000,␣
↪precision: 0.0000, hmean: 0.0000
08/21 21:45:59 - mmengine - INFO - prediction score threshold: 0.90, recall: 0.0000,␣
↪precision: 0.0000, hmean: 0.0000
08/21 21:45:59 - mmengine - INFO - Epoch(test) [10/10]  icdar/precision: 0.7222 ␣
↪icdar/recall: 0.6190  icdar/hmean: 0.6667
```

可以发现，模型在这个数据集上能达到的 hmean 为 0.6667，效果还是不错的。

小技巧： 若需要了解测试的高级用法，如 CPU 测试、多卡测试及集群测试等，请查阅训练与测试。

3.7 可视化输出

为了对模型的输出有一个更直观的感受，我们还可以直接可视化它的预测输出。在 `test.py` 中，用户可以通过 `show` 参数打开弹窗可视化；也可以通过 `show-dir` 参数指定预测结果图导出的目录。

```
python tools/test.py configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py␣
↪work_dirs/dbnet_r18_fpnc_1200e_icdar2015/epoch_400.pth --show-dir imgs/
```

真实标签和预测值会在可视化结果中以平铺的方式展示。左图的绿框表示真实标签，右图的红框表示预测值。

小技巧： 有关更多可视化功能的介绍，请参阅这里。

CHAPTER 4

推理

MMOCR 为示例和应用，以 ocr.py 脚本形式，提供了方便使用的 API。

该 API 可以通过命令行执行，也可以在 python 脚本内调用。在该 API 里，MMOCR 里的所有模型能以独立模块的形式被调用或串联。

警告: 该脚本仍在重构过程中，在接下来的版本接口中有可能会发生变化。

4.1 案例一：文本检测

注: 使用 TextSnake 检测模型对图像上的文本进行检测，并保存可视化的文件。

- 命令行执行:

```
python mmocr/ocr.py demo/demo_text_det.jpg --det TextSnake --img-out-dir demo/
```

- Python 调用:

```
from mmocr.ocr import MMOCR

# 导入模型到内存
ocr = MMOCR(det='TextSnake')
```

(下页继续)

(续上页)

```
# 推理
results = ocr.readtext('demo/demo_text_det.jpg', img_out_dir='demo/')
```

4.2 案例二：文本检测 + 识别

注：使用 DB_r18 检测模型和 CRNN 识别模型，对 demo/demo_text_det.jpg 图片执行 ocr（检测 + 识别）推理，在终端打印结果并展示可视化结果。

- 命令行执行：

```
python mmocr/ocr.py --det DB_r18 --recog CRNN demo/demo_text_ocr.jpg --print-result --
↪show
```

注解： 当用户从命令行执行脚本时，默认配置文件都会保存在 `configs/` 目录下。用户可以通过指定 `config_dir` 的值来自定义读取配置文件的文件夹。

- Python 调用：

```
from mmocr.ocr import MMOCR

# 导入模型到内存
ocr = MMOCR()

# 推理
results = ocr.readtext('demo/demo_text_ocr.jpg', print_result=True, show=True)
```

4.3 案例三：文本检测 + 识别 + 关键信息提取

注：首先，使用 DB_r18 检测模型和 CRNN 识别模型，进行端到端的 ocr （检测 + 识别）推理，然后对得到的结果，使用 SDMGR 模型提取关键信息（KIE），并展示可视化结果。

- 命令行执行：

```
python mmocr/ocr.py demo/demo_kie.jpeg  --det DB_r18 --recog CRNN --kie SDMGR --print-
↪result --show
```

注解： 当用户从命令行执行脚本时，默认配置文件都会保存在 `configs/` 目录下。用户可以通过指定 `config_dir` 的值来自定义读取配置文件的文件夹。

- Python 调用：

```
from mmocr.ocr import MMOCR

# 导入模型到内存
ocr = MMOCR(det='DB_r18', recog='CRNN', kie='SDMGR')

# 推理
results = ocr.readtext('demo/demo_kie.jpeg', print_result=True, show=True)
```

4.4 API 参数

该 API 有多个可供使用的参数列表。下表是 python 接口的参数。

MMOCR():

[1]: `kie` 当且仅当同时指定了文本检测和识别模型时才有效。

注解： mmocr 为了方便使用提供了预置的模型配置和对应的预训练权重，用户可以通过指定 `det` 和/或 `recog` 值来指定使用，这种方法等同于分别单独指定其对应的 `*_config` 和 `*_ckpt`。需要注意的是，手动指定 `*_config` 和 `*_ckpt` 会覆盖 `det` 和/或 `recog` 指定模型预置的配置和权重值。同理 `kie`,`kie_config` 和 `kie_ckpt` 的参数设定逻辑相同。

4.4.1 readtext()

以上所有参数在命令行同样适用，只需要在参数前简单添加两个连接符，并且将下参数中的下划线替换为连接符即可。（例如： `img_out_dir` 变成了 `--img-out-dir`）

对于布尔类型参数，添加在命令中默认为 true。（例如： `python mmocr/demo/ocr.py --det DB_r18 demo/demo_text_det.jpg --print_result` 意为 `print_result` 的参数值设置为 `True`）

4.5 模型

文本检测：

文本识别：

关键信息提取：

4.6 其他需要注意

- 执行检测 + 识别的推理（端到端 ocr），需要同时定义 `det` 和 `recog` 参数
- 如果只需要执行检测，则 `recog` 参数设置为 `None`。
- 如果只需要执行识别，则 `det` 参数设置为 `None`。

如果你对新特性有任何建议，请随时开一个 issue，甚至可以提一个 PR:)

CHAPTER 5

配置文档

MMOCR 主要使用 Python 文件作为配置文件。其配置文件系统的设计整合了模块化与继承的思想，方便用户进行各种实验。

5.1 常见用法

注解： 本小节建议结合 配置 (Config) 中的初级用法共同阅读。

MMOCR 最常用的操作为三种：配置文件的继承，对 _base_ 变量的引用以及对 _base_ 变量的修改。对于 _base_ 的继承与修改, MMEngine.Config 提供了两种语法，一种是针对 Python，Json，Yaml 均可使用的操作；另一种则仅适用于 Python 配置文件。在 MMOCR 中，我们**更推荐使用只针对 Python 的语法**，因此下文将以此为基础作进一步介绍。

这里以 `configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py` 为例，说明常用的三种用法。

```
_base_ = [
    '_base_dbnet_resnet18_fpnc.py',
    '../_base_/datasets/icdar2015.py',
    '../_base_/default_runtime.py',
    '../_base_/schedules/schedule_sgd_1200e.py',
]
```

(下页继续)

(续上页)

```
# dataset settings
ic15_det_train = _base_.ic15_det_train
ic15_det_train.pipeline = _base_.train_pipeline
ic15_det_test = _base_.ic15_det_test
ic15_det_test.pipeline = _base_.test_pipeline

train_dataloader = dict(
    batch_size=16,
    num_workers=8,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=True),
    dataset=ic15_det_train)

val_dataloader = dict(
    batch_size=1,
    num_workers=4,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=False),
    dataset=ic15_det_test)
```

5.1.1 配置文件的继承

配置文件存在继承的机制，即一个配置文件 A 可以将另一个配置文件 B 作为自己的基础并直接继承其中的所有字段，从而避免了大量的复制粘贴。

在 dbnet_resnet18_fpnc_1200e_icdar2015.py 中可以看到：

```
_base_ = [
    '_base_dbnet_resnet18_fpnc.py',
    '../_base_/datasets/icdar2015.py',
    '../_base_/default_runtime.py',
    '../_base_/schedules/schedule_sgd_1200e.py',
]
```

上述语句会读取列表中的所有基础配置文件，它们中的所有字段都会被载入到 dbnet_resnet18_fpnc_1200e_icdar2015.py 中。我们可以通过在 Python 解释中运行以下语句，了解配置文件被解析后的结构：

```
from mmengine import Config
db_config = Config.fromfile('configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_
↪icdar2015.py')
print(db_config)
```

可以发现，被解析的配置包含了所有 base 配置中的字段和信息。

注解： 请注意：各 *base* 配置文件中不能存在同名变量。

5.1.2 _base_ 变量的引用

有时，我们可能需要直接引用 _base_ 配置中的某些字段，以避免重复定义。假设我们想要获取 _base_ 配置中的变量 pseudo，就可以直接通过 _base_.pseudo 获得 _base_ 配置中的变量。

该语法已广泛用于 MMOCR 的配置中。MMOCR 中各个模型的数据集和管道（pipeline）配置都引用于基本配置。如在

```
ic15_det_train = _base_.ic15_det_train
# ...
train_dataloader = dict(
    # ...
    dataset=ic15_det_train)
```

5.1.3 _base_ 变量的修改

在 MMOCR 中不同算法在不同数据集通常有不同的数据流水线（pipeline），因此经常会会存在修改数据集中 pipeline 的场景。同时还存在很多场景需要修改 _base_ 配置中的变量，例如想修改某个算法的训练策略，某个模型的某些算法模块（更换 backbone 等）。用户可以直接利用 Python 的语法直接修改引用的 _base_ 变量。针对 dict，我们也提供了与类属性修改类似的方法，可以直接修改类属性修改字典内的内容。

1. 字典

   这里以修改数据集中的 pipeline 为例：

   可以利用 Python 语法修改字典：

   ```
   # 获取 _base_ 中的数据集
   ic15_det_train = _base_.ic15_det_train
   # 可以直接利用 Python 的 update 修改变量
   ic15_det_train.update(pipeline=_base_.train_pipeline)
   ```

   也可以使用类属性的方法进行修改：

   ```
   # 获取 _base_ 中的数据集
   ic15_det_train = _base_.ic15_det_train
   # 类属性方法修改
   ic15_det_train.pipeline = _base_.train_pipeline
   ```

2. 列表

   假设 _base_ 配置中的变量 pseudo = [1, 2, 3]，需要修改为 [1, 2, 4]:

```
# pseudo.py
pseudo = [1, 2, 3]
```

   可以直接重写:

```
_base_ = ['pseudo.py']
pseudo = [1, 2, 4]
```

   或者利用 Python 语法修改列表:

```
_base_ = ['pseudo.py']
pseudo = _base_.pseudo
pseudo[2] = 4
```

5.1.4 命令行修改配置

有时候我们只希望修部分配置，而不想修改配置文件本身。例如实验过程中想更换学习率，但是又不想重新写一个配置文件，可以通过命令行传入参数来覆盖相关配置。

我们可以在命令行里传入 --cfg-options，并在其之后的参数直接修改对应字段，例如我们想在运行 train 的时候修改学习率，只需要在命令行执行:

```
python tools/train.py example.py --cfg-options optim_wrapper.optimizer.lr=1
```

更多详细用法参考命令行修改配置

5.2 配置内容

通过配置文件与注册器的配合，MMOCR 可以在不侵入代码的前提下修改训练参数以及模型配置。具体而言，用户可以在配置文件中对如下模块进行自定义修改：环境配置、Hook 配置、日志配置、训练策略配置、数据相关配置、模型相关配置、评测配置、可视化配置。

本文档将以文字检测算法 DBNet 和文字识别算法 CRNN 为例来详细介绍 Config 中的内容。

5.2.1 环境配置

```
default_scope = 'mmocr'
env_cfg = dict(
    cudnn_benchmark=True,
    mp_cfg=dict(mp_start_method='fork', opencv_num_threads=0),
    dist_cfg=dict(backend='nccl'))
random_cfg = dict(seed=None)
```

主要包含三个部分：

- 设置所有注册器的默认 scope 为 mmocr，保证所有的模块首先从 MMOCR 代码库中进行搜索。若果该模块不存在，则继续从上游算法库 MMEngine 和 MMCV 中进行搜索（详见注册器。
- env_cfg 设置分布式环境配置，更多配置可以详见 MMEngine Runner
- random_cfg 设置 numpy，torch，cudnn 等随机种子，更多配置详见 Runner

5.2.2 Hook 配置

Hook 主要分为两个部分，默认 hook 以及自定义 hook。默认 hook 为所有任务想要运行所必须的配置，自定义 hook 一般服务于特定的算法或某些特定任务（目前为止 MMOCR 中没有自定义的 Hook）。

```
default_hooks = dict(
    timer=dict(type='IterTimerHook'), # 时间记录，包括数据增强时间以及模型推理时间
    logger=dict(type='LoggerHook', interval=1), # 日志打印间隔
    param_scheduler=dict(type='ParamSchedulerHook'), # 与 param_scheduler 更新学习率等超
参
    checkpoint=dict(type='CheckpointHook', interval=1),# 保存 checkpoint， interval 控制
保存间隔
    sampler_seed=dict(type='DistSamplerSeedHook'), # 多机情况下设置种子
    sync_buffer=dict(type='SyncBuffersHook'), # 同步多卡情况下，buffer
    visualization=dict( # 用户可视化 val 和 test 的结果
        type='VisualizationHook',
        interval=1,
        enable=False,
        show=False,
        draw_gt=False,
        draw_pred=False))
 custom_hooks = []
```

这里简单介绍几个经常可能会变动的 hook，通用的修改方法参考修改配置。

- LoggerHook：用于配置日志记录器的行为。例如，通过修改 interval 可以控制日志打印的间隔，每 interval 次迭代 (iteration) 打印一次日志，更多设置可参考 LoggerHook API。

- `CheckpointHook`：用于配置模型断点保存相关的行为，如保存最优权重，保存最新权重等。同样可以修改 `interval` 控制保存 checkpoint 的间隔。更多设置可参考 CheckpointHook API
- `VisualizationHook`：用于配置可视化相关行为，例如在验证或测试时可视化预测结果，默认为关。同时该 Hook 依赖可视化配置。想要了解详细功能可以参考*Visualizer*。更多配置可以参考*VisualizationHook API*。

如果想进一步了解默认 hook 的配置以及功能，可以参考钩子 (Hook)。

5.2.3 日志配置

此部分主要用来配置日志配置等级以及日志处理器。

```
log_level = 'INFO' # 日志记录等级
log_processor = dict(type='LogProcessor',
                         window_size=10,
                         by_epoch=True)
```

- 日志配置等级与 logging 的配置一致，
- 日志处理器主要用来控制输出的格式，详细功能可参考记录日志：
  - `by_epoch=True` 表示按照 epoch 输出日志，日志格式需要和 `train_cfg` 中的 `type='EpochBasedTrainLoop'` 参数保持一致。例如想按迭代次数输出日志，就需要令 `log_processor` 中的 `by_epoch=False` 的同时 `train_cfg` 中的 `type = 'IterBasedTrainLoop'`。
  - `window_size` 表示损失的平滑窗口，即最近 `window_size` 次迭代的各种损失的均值。logger 中最终打印的 loss 值为经过各种损失的平均值。

5.2.4 训练策略配置

此部分主要包含优化器设置、学习率策略和 `Loop` 设置。

对不同算法任务 (文字检测，文字识别，关键信息提取)，通常有自己任务常用的调参策略。这里列出了文字识别中的 `CRNN` 所用涉及的相应配置。

```
# 优化器
optim_wrapper = dict(
    type='OptimWrapper', optimizer=dict(type='Adadelta', lr=1.0))
param_scheduler = [dict(type='ConstantLR', factor=1.0)]
train_cfg = dict(type='EpochBasedTrainLoop',
                     max_epochs=5, # 训练轮数
                     val_interval=1) # 评测间隔
val_cfg = dict(type='ValLoop')
test_cfg = dict(type='TestLoop')
```

- optim_wrapper：主要包含两个部分，优化器封装 (OptimWrapper) 以及优化器 (Optimizer)。详情使用信息可见 MMEngine 优化器封装
  - 优化器封装支持不同的训练策略，包括混合精度训练 (AMP)、梯度累加和梯度截断。
  - 优化器设置中支持了 PyTorch 所有的优化器，所有支持的优化器见 PyTorch 优化器列表。
- param_scheduler：学习率调整策略，支持大部分 PyTorch 中的学习率调度器，例如 ExponentialLR，LinearLR，StepLR，MultiStepLR 等，使用方式也基本一致，所有支持的调度器见调度器接口文档，更多功能可以参考优化器参数调整策略
- train/test/val_cfg：任务的执行流程，MMEngine 提供了四种流程：EpochBasedTrainLoop，IterBasedTrainLoop，ValLoop，TestLoop 更多可以参考循环控制器。

5.2.5 数据相关配置

数据集配置

主要用于配置两个方向：

- 数据集的图像与标注文件的位置。
- 数据增强相关的配置。在 OCR 领域中，数据增强通常与模型强相关。

更多参数配置可以参考数据基类。

数据集字段的命名规则在 MMOCR 中为：

```
{数据集名称缩写}_{算法任务}_{训练/测试} = dict(...)
```

- 数据集缩写：见 数据集名称对应表
- 算法任务：文本检测-det，文字识别-rec，关键信息提取-kie
- 训练/测试：数据集用于训练还是测试

以识别为例，使用 Syn90k 作为训练集，以 icdar2013 和 icdar2015 作为测试集配置如下：

```
# 识别数据集配置
mj_rec_train = dict(
    type='OCRDataset',
    data_root='data/rec/Syn90k/',
    data_prefix=dict(img_path='mnt/ramdisk/max/90kDICT32px'),
    ann_file='train_labels.json',
    test_mode=False,
    pipeline=None)

ic13_rec_test = dict(
    type='OCRDataset',
```

(下页继续)

(续上页)

```
    data_root='data/rec/icdar_2013/',
    data_prefix=dict(img_path='Challenge2_Test_Task3_Images/'),
    ann_file='test_labels.json',
    test_mode=True,
    pipeline=None)

ic15_rec_test = dict(
    type='OCRDataset',
    data_root='data/rec/icdar_2015/',
    data_prefix=dict(img_path='ch4_test_word_images_gt/'),
    ann_file='test_labels.json',
    test_mode=True,
    pipeline=None)
```

数据流水线配置

MMOCR 中，数据集的构建与数据准备是相互解耦的。也就是说，OCRDataset 等数据集构建类负责完成标注文件的读取与解析功能；而数据变换方法（Data Transforms）则进一步实现了数据读取、数据增强、数据格式化等相关功能。

同时一般情况下训练和测试会存在不同的增强策略，因此一般会存在训练流水线（train_pipeline）和测试流水线（test_pipeline）。

训练流水线的数据增强流程通常为：数据读取 (LoadImageFromFile)-> 标注信息读取 (LoadXXXAnntation)-> 数据增强-> 数据格式化 (PackXXXInputs)。

测试流水线的数据增强流程通常为：数据读取 (LoadImageFromFile)-> 数据增强-> 标注信息读取 (LoadXXXAnntation)-> 数据格式化 (PackXXXInputs)。

更多信息可以参考数据流水线

由于 OCR 任务的特殊性，一般情况下不同模型有不同数据增强的方式，相同模型在不同数据集一般也会有不同的数据增强方式。以 CRNN 为例：

```
# 数据增强
file_client_args = dict(backend='disk')
train_pipeline = [
    dict(
        type='LoadImageFromFile',
        color_type='grayscale',
        file_client_args=dict(backend='disk'),
        ignore_empty=True,
        min_size=5),
    dict(type='LoadOCRAnnotations', with_text=True),
```

(下页继续)

(续上页)

```
    dict(type='Resize', scale=(100, 32), keep_ratio=False),
    dict(
        type='PackTextRecogInputs',
        meta_keys=('img_path', 'ori_shape', 'img_shape', 'valid_ratio'))
]
test_pipeline = [
    dict(
        type='LoadImageFromFile',
        color_type='grayscale',
        file_client_args=dict(backend='disk')),
    dict(
        type='RescaleToHeight',
        height=32,
        min_width=32,
        max_width=None,
        width_divisor=16),
    dict(type='LoadOCRAnnotations', with_text=True),
    dict(
        type='PackTextRecogInputs',
        meta_keys=('img_path', 'ori_shape', 'img_shape', 'valid_ratio'))
]
```

Dataloader 配置

主要为构造数据集加载器 (dataloader) 所需的配置信息，更多教程看参考 PyTorch 数据加载器。

```
# Dataloader 部分
train_dataloader = dict(
    batch_size=64,
    num_workers=8,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=True),
    dataset=dict(
        type='ConcatDataset',
        datasets=[mj_rec_train],
        pipeline=train_pipeline))
val_dataloader = dict(
    batch_size=1,
    num_workers=4,
    persistent_workers=True,
    drop_last=False,
    sampler=dict(type='DefaultSampler', shuffle=False),
    dataset=dict(
```

(下页继续)

(续上页)

```
        type='ConcatDataset',
        datasets=[ic13_rec_test, ic15_rec_test],
        pipeline=test_pipeline))
test_dataloader = val_dataloader
```

5.2.6 模型相关配置

网络配置

用于配置模型的网络结构，不同的算法任务有不同的网络结构，

文本检测

文本检测主要包含几个部分:

- data_preprocessor: 数据处理器
- backbone: 特征提取网络
- neck: 颈网络配置
- det_head: 检测头网络配置
  - module_loss: 模型损失函数配置
  - postprocessor: 模型预测结果后处理配置

我们以 DBNet 为例，介绍文字检测中模型配置:

```
model = dict(
    type='DBNet',
    data_preprocessor=dict(
        type='TextDetDataPreprocessor',
        mean=[123.675, 116.28, 103.53],
        std=[58.395, 57.12, 57.375],
        bgr_to_rgb=True,
        pad_size_divisor=32)
    backbone=dict(
        type='mmdet.ResNet',
        depth=18,
        num_stages=4,
        out_indices=(0, 1, 2, 3),
        frozen_stages=-1,
        norm_cfg=dict(type='BN', requires_grad=True),
        init_cfg=dict(type='Pretrained', checkpoint='torchvision://resnet18'),
```

(下页继续)

(续上页)

```
        norm_eval=False,
        style='caffe'),
    neck=dict(
        type='FPNC', in_channels=[64, 128, 256, 512], lateral_channels=256),
    det_head=dict(
        type='DBHead',
        in_channels=256,
        module_loss=dict(type='DBModuleLoss'),
        postprocessor=dict(type='DBPostprocessor', text_repr_type='quad')))
```

文本识别

文本识别主要包含：

- data_processor: 数据预处理配置
- preprocessor: 网络预处理配置，如 TPS 等
- backbone: 特征提取配置
- encoder: 编码器配置
- decoder: 解码器配置
  - module_loss: 解码器损失
  - postprocessor: 解码器后处理
  - dictionary: 字典配置

以 CRNN 为例：

```
# 模型部分
model = dict(
   type='CRNN',
   data_preprocessor=dict(
        type='TextRecogDataPreprocessor', mean=[127], std=[127])
    preprocessor=None,
    backbone=dict(type='VeryDeepVgg', leaky_relu=False, input_channels=1),
    encoder=None,
    decoder=dict(
        type='CRNNDecoder',
        in_channels=512,
        rnn_flag=True,
        module_loss=dict(type='CTCModuleLoss', letter_case='lower'),
        postprocessor=dict(type='CTCPostProcessor'),
        dictionary=dict(
```

(下页继续)

(续上页)

```
            type='Dictionary',
            dict_file='dicts/lower_english_digits.txt',
            with_padding=True)))
```

权重加载配置

可以通过 load_from 参数加载检查点（checkpoint）文件中的模型权重，只需要将 load_from 参数设置为检查点文件的路径即可。

用户也可通过设置 resume=True ，加载检查点中的训练状态信息来恢复训练。当 load_from 和 resume=True 同时被设置时，执行器将加载 load_from 路径对应的检查点文件中的训练状态。

如果仅设置 resume=True，执行器将会尝试从 work_dir 文件夹中寻找并读取最新的检查点文件

```
load_from = None # 加载 checkpoint 的路径
resume = False # 是否 resume
```

更多可以参考加载权重或恢复训练与OCR 进阶技巧-断点恢复训练。

5.2.7 评测配置

在模型验证和模型测试中，通常需要对模型精度做定量评测。MMOCR 通过评测指标 (Metric) 和评测器 (Evaluator) 来完成这一功能。更多可以参考评测指标（Metric）和评测器（Evaluator）

评测部分包含两个部分，评测器和评测指标。接下来我们分部分展开讲解。

评测器

评测器主要用来管理多个数据集以及多个 Metric。针对单数据集与多数据集情况，评测器分为了单数据集评测器与多数据集评测器，这两种评测器均可管理多个 Metric.

单数据集评测器配置如下：

```
# 单个数据集 单个 Metric 情况
val_evaluator = dict(
    type='Evaluator',
    metrics=dict())

# 单个数据集 多个 Metric 情况
val_evaluator = dict(
    type='Evaluator',
    metrics=[...])
```

在实现中默认为单数据集评测器，因此对单数据集评测情况下，一般情况下只需配置评测器，即为

```
# 单个数据集 单个 Metric 情况
val_evaluator = dict()

# 单个数据集 多个 Metric 情况
val_evaluator = [...]
```

多数据集评测与单数据集评测存在两个位置上的不同：评测器类别与前缀。评测器类别必须为 MultiDatasetsEvaluator 且不能省略，前缀主要用来区分不同数据集在相同评测指标下的结果，请参考多数据集评测。

假设我们需要在 IC13 和 IC15 情况下测试精度，则配置如下：

```
# 多个数据集，单个 Metric 情况
val_evaluator = dict(
    type='MultiDatasetsEvaluator',
    metrics=dict(),
    dataset_prefixes=['IC13', 'IC15'])

# 多个数据集，多个 Metric 情况
val_evaluator = dict(
    type='MultiDatasetsEvaluator',
    metrics=[...],
    dataset_prefixes=['IC13', 'IC15'])
```

评测指标

评测指标指不同度量精度的方法，同时可以多个评测指标共同使用，更多评测指标原理参考评测指标，在 MMOCR 中不同算法任务有不同的评测指标。

文字检测: HmeanIOU

文字识别: WordMetric，CharMetric，OneMinusNEDMetric

关键信息提取: F1Metric

以文本检测为例说明，在单数据集评测情况下，使用单个 Metric：

```
val_evaluator = dict(type='HmeanIOUMetric')
```

以文本识别为例，多数据集使用多个 Metric 评测：

```
# 评测部分
val_evaluator = dict(
    type='MultiDatasetsEvaluator',
    metrics=[
        dict(
```

(下页继续)

(续上页)

```
            type='WordMetric',
            mode=['exact', 'ignore_case', 'ignore_case_symbol']),
        dict(type='CharMetric')
    ],
    dataset_prefixes=['IC13', 'IC15'])
test_evaluator = val_evaluator
```

5.2.8 可视化配置

每个任务配置该任务对应的可视化器。可视化器主要用于用户模型中间结果的可视化或存储，及 val 和 test 预测结果的可视化。同时可视化的结果可以通过可视化后端储存到不同的后端，比如 Wandb，TensorBoard 等。常用修改操作可见可视化。

文本检测的可视化默认配置如下：

```
vis_backends = [dict(type='LocalVisBackend')]
visualizer = dict(
    type='TextDetLocalVisualizer',  # 不同任务有不同的可视化器
    vis_backends=vis_backends,
    name='visualizer')
```

5.3 目录结构

MMOCR 所有配置文件都放置在 configs 文件夹下。为了避免配置文件过长，同时提高配置文件的可复用性以及清晰性，MMOCR 利用 Config 文件的继承特性，将配置内容的八个部分做了拆分。因为每部分均与算法任务相关，因此 MMOCR 对每个任务在 Config 中提供了一个任务文件夹，即 textdet (文字检测任务)、textrec (文字识别任务)、kie (关键信息提取)。同时各个任务算法配置文件夹下进一步划分为两个部分：_base_ 文件夹与诸多算法文件夹：

1. _base_ 文件夹下主要存放与具体算法无关的一些通用配置文件，各部分依目录分为常用的数据集、常用的训练策略以及通用的运行配置。
2. 算法配置文件夹中存放与算法强相关的配置项。算法配置文件夹主要分为两部分：
    1. 算法的模型与数据流水线：OCR 领域中一般情况下数据增强策略与算法强相关，因此模型与数据流水线通常置于统一位置。
    2. 算法在制定数据集上的特定配置：用于训练和测试的配置，将分散在不同位置的配置汇总。同时修改或配置一些在该数据集特有的配置比如 batch size 以及一些可能修改如数据流水线，训练策略等

最后的将配置内容中的各个模块分布在不同配置文件中，最终各配置文件内容如下：

最终目录结构如下：

```
config
├── textdet
│   ├── _base_
│   │   ├── datasets
│   │   │   ├── icdar2015.py
│   │   │   ├── icdar2017.py
│   │   │   └── totaltext.py
│   │   ├── schedules
│   │   │   └── schedule_adam_600e.py
│   │   └── default_runtime.py
│   └── dbnet
│       ├── _base_dbnet_resnet18_fpnc.py
│       └── dbnet_resnet18_fpnc_1200e_icdar2015.py
├── textrecog
│   ├── _base_
│   │   ├── datasets
│   │   │   ├── icdar2015.py
│   │   │   ├── icdar2017.py
│   │   │   └── totaltext.py
│   │   ├── schedules
│   │   │   └── schedule_adam_base.py
│   │   └── default_runtime.py
│   └── crnn
│       ├── _base_crnn_mini-vgg.py
│       └── crnn_mini-vgg_5e_mj.py
└── kie
    ├── _base_
    │   ├──datasets
    │   └── default_runtime.py
    └── sgdmr
        └── sdmgr_novisual_60e_wildreceipt_openset.py
```

5.4 配置文件以及权重命名规则

MMOCR 按照以下风格进行配置文件命名，代码库的贡献者需要遵循相同的命名规则。文件名总体分为四部分：算法信息，模块信息，训练信息和数据信息。逻辑上属于不同部分的单词之间用下划线 `'_'` 连接，同一部分有多个单词用短横线 `'-'` 连接。

```
{{算法信息}}_{{模块信息}}_{{训练信息}}_{{数据信息}}.py
```

- 算法信息 (algorithm info)：算法名称，如 DBNet，CRNN 等
- 模块信息 (module info)：按照数据流的顺序列举一些中间的模块，其内容依赖于算法任务，同时为了避

免 Config 过长，会省略一些与模型强相关的模块。下面举例说明：

  - 对于文字检测任务和关键信息提取任务：

```
{{算法信息}}_{{backbone}}_{{neck}}_{{head}}_{{训练信息}}_{{数据信息}}.py
```

    一般情况下 head 位置一般为算法专有的 head，因此一般省略。

  - 对于文本识别任务：

```
{{算法信息}}_{{backbone}}_{{encoder}}_{{decoder}}_{{训练信息}}_{{数据信息}}.py
```

    一般情况下 encode 和 decoder 位置一般为算法专有，因此一般省略。

- 训练信息 (training info)：训练策略的一些设置，包括 batch size，schedule 等
- 数据信息 (data info)：数据集名称、模态、输入尺寸等，如 icdar2015，synthtext 等

CHAPTER 6

数据集准备

6.1 前言

经过数十年的发展，OCR 领域涌现出了一系列的相关数据集，这些数据集往往采用风格各异的格式来提供文本的标注文件，使得用户在使用这些数据集时不得不进行格式转换。MMOCR 支持了数十种常用的文本相关数据集，并提供了详细的数据下载及准备教程。

另外，我们为各任务常用的数据集提供了数据格式转换脚本，以帮助用户快速将数据转换为 MMOCR 支持的格式。

- 文本检测数据集准备
- 文本识别数据集准备
- 关键信息抽取数据集准备

下面，我们对 MMOCR 内支持的各任务的数据格式进行简要的介绍。

- 如以下代码块所示，文本检测任务采用数据格式 `TextDetDataset`，其中存放了文本检测任务所需的边界盒标注、文件名等信息。我们在 `tests/data/det_toy_dataset/instances_test.json` 路径中提供了一个示例标注文件。

```
{
"metainfo":
  {
    "dataset_type": "TextDetDataset",
    "task_name": "textdet",
```

(下页继续)

(续上页)

```
    "category": [{"id": 0, "name": "text"}]
  },
 "data_list":
  [
    {
      "img_path": "test_img.jpg",
      "height": 640,
      "width": 640,
      "instances":
        [
          {
            "polygon": [0, 0, 0, 10, 10, 20, 20, 0],
            "bbox": [0, 0, 10, 20],
            "bbox_label": 0,
            "ignore": false
          }
        ],
        //...
    }
  ]
}
```

- 如以下代码块所示，文本识别任务采用数据格式 `TextRecogDataset`，其中存放了文本识别任务所需的文本内容及图片路径等信息。我们在 `tests/data/rec_toy_dataset/labels.json` 路径中提供了一个示例标注文件。

```
{
 "metainfo":
   {
     "dataset_type": "TextRecogDataset",
     "task_name": "textrecog",
   },
 "data_list":
   [
     {
       "img_path": "test_img.jpg",
       "instances":
         [
           {
             "text": "GRAND"
           }
         ]
       }
```

(下页继续)

(续上页)

```
    ]
}
```

6.2 数据集下载及格式转换

以 ICDAR 2015 **文本检测数据集**的准备步骤为例，你可以依次执行以下步骤来完成数据集准备：

- 从 ICDAR 官方网站下载 ICDAR 2015 数据集。将训练集 `ch4_training_word_images_gt.zip` 与测试集压缩包 `ch4_test_word_images_gt.zip` 分别解压至路径 `data/icdar2015`。

```
# 下载数据集
mkdir data/det/icdar2015 && cd data/det/icdar2015
wget https://rrc.cvc.uab.es/downloads/ch4_training_images.zip --no-check-
↪certificate
wget https://rrc.cvc.uab.es/downloads/ch4_training_localization_transcription_gt.
↪zip --no-check-certificate
wget https://rrc.cvc.uab.es/downloads/ch4_test_images.zip --no-check-certificate
wget https://rrc.cvc.uab.es/downloads/Challenge4_Test_Task1_GT.zip --no-check-
↪certificate

# 解压数据集
mkdir imgs && mkdir annotations
unzip ch4_training_images.zip -d imgs/training
unzip ch4_training_localization_transcription_gt.zip -d annotations/training
unzip ch4_test_images.zip -d imgs/test
unzip Challenge4_Test_Task1_GT.zip -d annotations/test
```

- 使用 MMOCR 提供的格式转换脚本将原始的标注文件转换为 MMOCR 统一的数据格式

```
python tools/dataset_converters/textdet/icdar_converter.py data/det/icdar15/ -o␣
↪data/det/icdar15/ --split-list training test -d icdar2015
```

- 完成上述步骤后，数据集标签将被转换为 MMOCR 使用的统一格式，文件目录结构如下：

```
data/det/icdar2015/
├── annotations
│   ├── test
│   └── training
├── imgs
│   ├── test
│   └── training
├── instances_test.json
└── instances_training.json
```

6.3 数据集配置文件

6.3.1 单数据集训练

在使用新的数据集时，我们需要对其图像、标注文件的路径等基础信息进行配置。configs/xxx/_base_/datasets/ 路径下已预先配置了 MMOCR 中常用的数据集，这里我们以 ICDAR 2015 数据集为例（见 configs/_base_/det_datasets/icdar2015.py）：

```
ic15_det_data_root = 'data/det/icdar2015' # 数据集根目录

# 训练集配置
ic15_det_train = dict(
    type='OCRDataset',
    data_root=ic15_det_data_root,                          # 数据根目录
    ann_file='instances_training.json',                    # 标注文件名称
    data_prefix=dict(img_path='imgs/'),                    # 图片路径前缀
    filter_cfg=dict(filter_empty_gt=True, min_size=32),    # 数据过滤
    pipeline=None)
# 测试集配置
ic15_det_test = dict(
    type='OCRDataset',
    data_root=ic15_det_data_root,
    ann_file='instances_test.json',
    data_prefix=dict(img_path='imgs/'),
    test_mode=True,
    pipeline=None)
```

在配置好数据集后，我们还需要在相应的算法模型配置文件中导入想要使用的数据集。例如，在 ICDAR 2015 数据集上训练“DBNet_R18”模型：

```
_base_ = [
    '_base_dbnet_r18_fpnc.py',
    '../_base_/datasets/icdar2015.py',  # 导入数据集配置文件
    '../_base_/default_runtime.py',
    '../_base_/schedules/schedule_sgd_1200e.py',
]

ic15_det_train = _base_.ic15_det_train              # 指定训练集
ic15_det_train.pipeline = _base_.train_pipeline     # 指定训练集使用的数据流水线
ic15_det_test = _base_.ic15_det_test                # 指定测试集
ic15_det_test.pipeline = _base_.test_pipeline       # 指定测试集使用的数据流水线

train_dataloader = dict(
    batch_size=16,
```

(下页继续)

(续上页)

```
    num_workers=8,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=True),
    dataset=ic15_det_train)    # 在 train_dataloader 中指定使用的训练数据集

val_dataloader = dict(
    batch_size=1,
    num_workers=4,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=False),
    dataset=ic15_det_test)    # 在 val_dataloader 中指定使用的验证数据集

test_dataloader = val_dataloader
```

6.3.2 多数据集训练

此外，基于*ConcatDataset*，用户还可以使用多个数据集组合来训练或测试模型。用户只需在配置文件中将 dataloader 中的 dataset 类型设置为 `ConcatDataset`，并指定对应的数据集列表即可。

```
train_list = [ic11, ic13, ic15]
train_dataloader = dict(
    dataset=dict(
        type='ConcatDataset', datasets=train_list, pipeline=train_pipeline))
```

例如，以下配置使用了 MJSynth 数据集进行训练，并使用 6 个学术数据集（CUTE80, IIIT5K, SVT, SVTP, ICDAR2013, ICDAR2015）进行测试。

```
_base_ = [ # 导入所有需要使用的数据集配置
    '../_base_/datasets/mjsynth.py',
    '../_base_/datasets/cute80.py',
    '../_base_/datasets/iiit5k.py',
    '../_base_/datasets/svt.py',
    '../_base_/datasets/svtp.py',
    '../_base_/datasets/icdar2013.py',
    '../_base_/datasets/icdar2015.py',
    '../_base_/default_runtime.py',
    '../_base_/schedules/schedule_adadelta_5e.py',
    '_base_crnn_mini-vgg.py',
]

# 训练集列表
train_list = [_base_.mj_rec_train]
```

(下页继续)

(续上页)

```
# 测试集列表
test_list = [
    _base_.cute80_rec_test, _base_.iiit5k_rec_test, _base_.svt_rec_test,
    _base_.svtp_rec_test, _base_.ic13_rec_test, _base_.ic15_rec_test
]

# 使用 ConcatDataset 来级联列表中的多个数据集
train_dataset = dict(
       type='ConcatDataset', datasets=train_list, pipeline=_base_.train_pipeline)
test_dataset = dict(
       type='ConcatDataset', datasets=test_list, pipeline=_base_.test_pipeline)

train_dataloader = dict(
    batch_size=192 * 4,
    num_workers=32,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=True),
    dataset=train_dataset)

test_dataloader = dict(
    batch_size=1,
    num_workers=4,
    persistent_workers=True,
    drop_last=False,
    sampler=dict(type='DefaultSampler', shuffle=False),
    dataset=test_dataset)

val_dataloader = test_dataloader
```

CHAPTER 7

训练与测试

为了适配多样化的用户需求，MMOCR 实现了多种不同操作系统及设备上的模型训练及测试。无论是使用本地机器进行单机单卡训练测试，还是在部署了 slurm 系统的大规模集群上进行训练测试，MMOCR 都提供了便捷的解决方案。

7.1 单卡机器训练及测试

7.1.1 训练

`tools/train.py` 实现了基础的训练服务。MMOCR 推荐用户使用 GPU 进行模型训练和测试，但是，用户也可以通过指定 `CUDA_VISIBLE_DEVICES=-1` 来使用 CPU 设备进行模型训练及测试。例如，以下命令演示了如何使用 CPU 或单卡 GPU 来训练 DBNet 文本检测器。

```
# 通过调用 tools/train.py 来训练指定的 MMOCR 模型
CUDA_VISIBLE_DEVICES= python tools/train.py ${CONFIG_FILE} [PY_ARGS]

# 训练
# 示例 1: 使用 CPU 训练 DBNet
CUDA_VISIBLE_DEVICES=-1 python tools/train.py configs/textdet/dbnet/dbnet_resnet50-
↪dcnv2_fpnc_1200e_icdar2015.py

# 示例 2: 指定使用 gpu:0 训练 DBNet，指定工作目录为 dbnet/，并打开混合精度 (amp) 训练
CUDA_VISIBLE_DEVICES=0 python tools/train.py configs/textdet/dbnet/dbnet_resnet50-
↪dcnv2_fpnc_1200e_icdar2015.py --work-dir dbnet/ --amp
```

注解: 此外，如需使用指定编号的 GPU 进行训练或测试，例如使用 3 号 GPU，则可以通过设定 CUDA_VISIBLE_DEVICES=3 来实现。

下表列出了 train.py 支持的所有参数。其中，不带 -- 前缀的参数为必须的位置参数，带 -- 前缀的参数为可选参数。

7.1.2 测试

tools/test.py 提供了基础的测试服务，其使用原理和训练脚本类似。例如，以下命令演示了 CPU 或 GPU 单卡测试 DBNet 模型。

```
# 通过调用 tools/test.py 来测试指定的 MMOCR 模型
CUDA_VISIBLE_DEVICES= python tools/test.py ${CONFIG_FILE} ${CHECKPOINT_FILE} [PY_ARGS]

# 测试
# 示例 1: 使用 CPU 测试 DBNet
CUDA_VISIBLE_DEVICES=-1 python tools/test.py configs/textdet/dbnet/dbnet_resnet50-
↪dcnv2_fpnc_1200e_icdar2015.py dbnet_r50.pth
# 示例 2: 使用 gpu:0 测试 DBNet
CUDA_VISIBLE_DEVICES=0 python tools/test.py configs/textdet/dbnet/dbnet_resnet50-
↪dcnv2_fpnc_1200e_icdar2015.py dbnet_r50.pth
```

下表列出了 test.py 支持的所有参数。其中，不带 -- 前缀的参数为必须的位置参数，带 -- 前缀的参数为可选参数。

7.2 多卡机器训练及测试

对于大规模模型，采用多 GPU 训练和测试可以极大地提升操作的效率。为此，MMOCR 提供了基于 MMDistributedDataParallel 实现的分布式脚本 tools/dist_train.sh 和 tools/dist_test.sh。

```
# 训练
NNODES=${NNODES} NODE_RANK=${NODE_RANK} PORT=${MASTER_PORT} MASTER_ADDR=${MASTER_ADDR}
↪ ./tools/dist_train.sh ${CONFIG_FILE} ${GPU_NUM} [PY_ARGS]
# 测试
NNODES=${NNODES} NODE_RANK=${NODE_RANK} PORT=${MASTER_PORT} MASTER_ADDR=${MASTER_ADDR}
↪ ./tools/dist_test.sh ${CONFIG_FILE} ${CHECKPOINT_FILE} ${GPU_NUM} [PY_ARGS]
```

下表列出了 dist_*.sh 支持的参数：

这两个脚本可以实现**单机多卡**或**多机多卡**的训练和测试，下面演示了它们在不同场景下的用法。

7.2.1 单机多卡

以下命令演示了如何在搭载多块 GPU 的**单台机器**上使用指定数目的 GPU 进行训练及测试：

1. **训练**

   使用单台机器上的 4 块 GPU 训练 DBNet。

```
# 单机 4 卡训练 DBNet
tools/dist_train.sh configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py 4
```

2. **测试**

   使用单台机器上的 4 块 GPU 测试 DBNet。

```
# 单机 4 卡测试 DBNet
tools/dist_test.sh configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py␣
↪dbnet_r50.pth 4
```

7.2.2 单机多任务训练及测试

对于搭载多块 GPU 的单台服务器而言，用户可以通过指定 GPU 的形式来同时执行不同的训练任务。例如，以下命令演示了如何在一台 8 卡 GPU 服务器上分别使用 `[0, 1, 2, 3]` 卡测试 DBNet 及 `[4, 5, 6, 7]` 卡训练 CRNN：

```
# 指定使用 gpu:0,1,2,3 测试 DBNet，并分配端口号 29500
CUDA_VISIBLE_DEVICES=0,1,2,3 PORT=29500 ./tools/dist_test.sh configs/textdet/dbnet/
↪dbnet_r50dcnv2_fpnc_1200e_icdar2015.py dbnet_r50.pth 4
# 指定使用 gpu:4,5,6,7 训练 CRNN，并分配端口号 29501
CUDA_VISIBLE_DEVICES=4,5,6,7 PORT=29501 ./tools/dist_train.sh configs/textrecog/crnn/
↪crnn_academic_dataset.py 4
```

注解： `dist_train.sh` 默认将 `MASTER_PORT` 设置为 `29500`，当单台机器上有其它进程已占用该端口时，程序则会出现运行时错误 `RuntimeError: Address already in use`。此时，用户需要将 `MASTER_PORT` 设置为 `(0~65535)` 范围内的其它空闲端口号。

7.2.3 多机多卡训练及测试

MMOCR 基于torch.distributed 提供了相同局域网下的多台机器间的多卡分布式训练。

1. **训练**

   以下命令演示了如何在两台机器上分别使用 2 张 GPU 合计 4 卡训练 DBNet：

```
# 示例：在两台机器上分别使用 2 张 GPU 合计 4 卡训练 DBNet
# 在 “机器 1” 上运行以下命令
NNODES=2 NODE_RANK=0 PORT=29501 MASTER_ADDR=10.140.0.169 tools/dist_train.sh␣
↪configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py 2
# 在 “机器 2” 上运行以下命令
NNODES=2 NODE_RANK=1 PORT=29501 MASTER_ADDR=10.140.0.169 tools/dist_train.sh␣
↪configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py 2
```

2. **测试**

   以下命令演示了如何在两台机器上分别使用 2 张 GPU 合计 4 卡测试：

```
# 示例：在两台机器上分别使用 2 张 GPU 合计 4 卡测试
# 在 “机器 1” 上运行以下命令
NNODES=2 NODE_RANK=0 PORT=29500 MASTER_ADDR=10.140.0.169 tools/dist_test.sh␣
↪configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py dbnet_r50.pth 2
# 在 “机器 2” 上运行以下命令
NNODES=2 NODE_RANK=1 PORT=29501 MASTER_ADDR=10.140.0.169 tools/dist_test.sh␣
↪configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py dbnet_r50.pth 2
```

注解： 需要注意的是，采用多机多卡训练时，机器间的网络传输速度可能成为训练速度的瓶颈。

7.3 集群训练及测试

针对 Slurm 调度系统管理的计算集群，MMOCR 提供了对应的训练和测试任务提交脚本 `tools/slurm_train.sh` 及 `tools/slurm_test.sh`。

```
# tools/slurm_train.sh 提供基于 slurm 调度系统管理的计算集群上提交训练任务的脚本
GPUS=${GPUS} GPUS_PER_NODE=${GPUS_PER_NODE} CPUS_PER_TASK=${CPUS_PER_TASK} SRUN_ARGS=$
↪{SRUN_ARGS} ./tools/slurm_train.sh ${PARTITION} ${JOB_NAME} ${CONFIG_FILE} ${WORK_
↪DIR} [PY_ARGS]

# tools/slurm_test.sh 提供基于 slurm 调度系统管理的计算集群上提交测试任务的脚本
GPUS=${GPUS} GPUS_PER_NODE=${GPUS_PER_NODE} CPUS_PER_TASK=${CPUS_PER_TASK} SRUN_ARGS=$
↪{SRUN_ARGS} ./tools/slurm_test.sh ${PARTITION} ${JOB_NAME} ${CONFIG_FILE} $
↪{CHECKPOINT_FILE} ${WORK_DIR} [PY_ARGS]
```

(下页继续)

(续上页)

这两个脚本可以实现 slurm 集群上的训练和测试，下面演示了它们在不同场景下的用法。

1. 训练

   以下示例为在 slurm 集群 dev 分区申请 1 块 GPU 进行 DBNet 训练。

```
# 示例：在 slurm 集群 dev 分区申请 1 块 GPU 资源进行 DBNet 训练任务
GPUS=1 GPUS_PER_NODE=1 CPUS_PER_TASK=5 tools/slurm_train.sh dev db_r50 configs/
↪textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py work_dir
```

2. 测试

   同理，则提供了测试任务提交脚本。以下示例为在 slurm 集群 dev 分区申请 1 块 GPU 资源进行 DBNet 测试。

```
# 示例：在 slurm 集群 dev 分区申请 1 块 GPU 资源进行 DBNet 测试任务
GPUS=1 GPUS_PER_NODE=1 CPUS_PER_TASK=5 tools/slurm_test.sh dev db_r50 configs/textdet/
↪dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py dbnet_r50.pth work_dir
```

7.4 进阶技巧

7.4.1 从断点恢复训练

`tools/train.py` 提供了从断点恢复训练的功能，用户仅需在命令中指定 `--resume` 参数，即可自动从断点恢复训练。

```
# 示例：从断点恢复训练
python tools/train.py configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py 4 -
↪-resume
```

默认地，程序将自动从上次训练过程中最后成功保存的断点，即 `latest.pth` 处开始继续训练。如果用户希望指定从特定的断点处开始恢复训练，则可以按如下格式在模型的配置文件中设定该断点的路径。

```
# 示例：在配置文件中设置想要加载的断点路径
load_from = 'work_dir/dbnet/models/epoch_10000.pth'
```

7.4.2 混合精度训练

混合精度训练可以在缩减内存占用的同时提升训练速度，为此，MMOCR 提供了一键式的混合精度训练方案，仅需在训练时添加 `--amp` 参数即可。

```
# 示例：使用自动混合精度训练
python tools/train.py configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py 4 -
↪-amp
```

下表列出了 MMOCR 中各算法对自动混合精度训练的支持情况：

7.4.3 自动学习率缩放

MMOCR 在配置文件中为每一个模型设置了默认的初始学习率，然而，当用户使用的 `batch_size` 不同于我们预设的 `base_batch_size` 时，这些初始学习率可能不再完全适用。因此，我们提供了自动学习率缩放工具。当使用不同于 MMOCR 预设的 `base_batch_size` 进行训练时，用户仅需添加 `--auto-scale-lr` 参数即可自动依据新的 `batch_size` 将学习率缩放至对应尺度。

```
# 示例：使用自动学习率缩放
python tools/train.py configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py 4 -
↪-auto-scale-lr
```

7.4.4 可视化模型测试结果

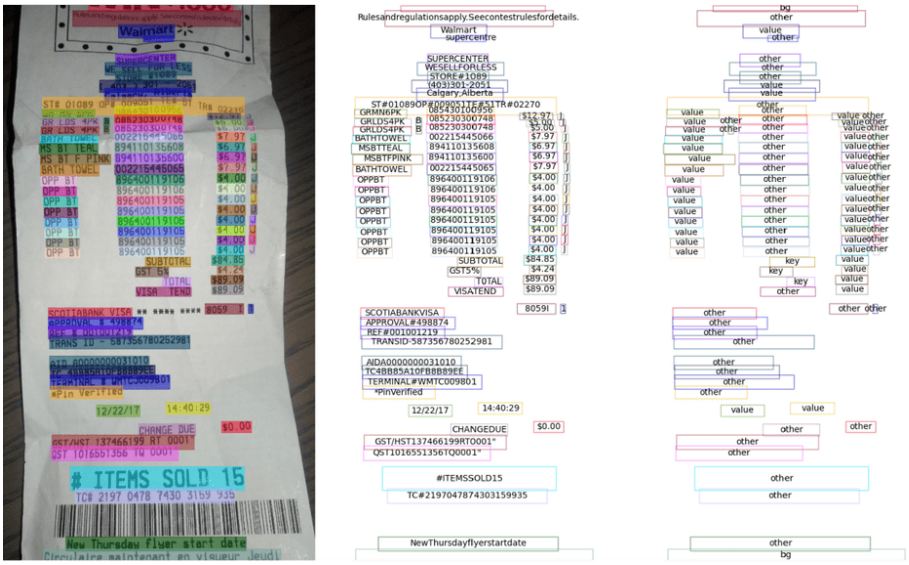
`tools/test.py` 提供了可视化接口，以方便用户对模型进行定性分析。

(绿色框为真实标注，红色框为预测结果)

GABBANA

GABBANA

（绿色字体为真实标注，红色字体为预测结果）



（从左至右分别为：原图，文本检测和识别结果，文本分类结果，关系图）

```
# 示例 1: 每间隔 2 秒绘制出
python tools/test.py configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py␣
↪dbnet_r50.pth --show --wait-time 2

# 示例 2: 对于不支持图形化界面的系统 (如计算集群等), 可以将可视化结果存入指定路径
python tools/test.py configs/textdet/dbnet/dbnet_r50dcnv2_fpnc_1200e_icdar2015.py␣
↪dbnet_r50.pth --show-dir ./vis_results
```

tools/test.py 中可视化相关参数说明：

CHAPTER 8

可视化

阅读本文前建议先阅读 MMEngine 的可视化 (Visualization)文档以初步了解 Visualizer 的定义及相关用法。

简单来说，MMEngine 中实现了用于满足日常可视化需求的可视化器件 `Visualizer`，其主要包含三个功能：

- 实现了常用的绘图 API，例如 `draw_bboxes` 实现了边界盒的绘制功能，`draw_lines` 实现了线条的绘制功能。
- 支持将可视化结果、学习率曲线、损失函数曲线以及验证精度曲线等写入多种后端中，包括本地磁盘以及常用的深度学习训练日志记录工具，如 TensorBoard 和 WandB。
- 支持在代码中的任意位置进行调用，例如在训练或测试过程中可视化或记录模型的中间状态，如特征图及验证结果等。

基于 MMEngine 的 Visualizer，MMOCR 内预置了多种可视化工具，用户仅需简单修改配置文件即可使用：

- `tools/analysis_tools/browse_dataset.py` 脚本提供了数据集可视化功能，其可以绘制经过数据变换（Data Transforms）之后的图像及对应的标注内容，详见*browse_dataset.py*。
- MMEngine 中实现了 `LoggerHook`，该 Hook 利用 `Visualizer` 将学习率、损失以及评估结果等数据写入 `Visualizer` 设置的后端中，因此通过修改配置文件中的 `Visualizer` 后端，比如修改为 `TensorBoardVISBackend` 或 `WandbVISBackend`，可以实现将日志到 `TensorBoard` 或 `WandB` 等常见的训练日志记录工具中，从而方便用户使用这些可视化工具来分析和监控训练流程。
- MMOCR 中实现了 `VisualizerHook`，该 Hook 利用 `Visualizer` 将验证阶段或预测阶段的预测结果进行可视化或储存至 `Visualizer` 设置的后端中，因此通过修改配置文件中的 `Visualizer` 后端，比如修改为 `TensorBoardVISBackend` 或 `WandbVISBackend`，可以实现将预测的图像存储到 `TensorBoard` 或 `Wandb` 中。

8.1 配置

得益于注册机制的使用，在 MMOCR 中，我们可以通过修改配置文件来设置可视化器件 Visualizer 的行为。通常，我们在 task/_base_/default_runtime.py 中定义可视化相关的默认配置，详见配置教程。

```
vis_backends = [dict(type='LocalVisBackend')]
visualizer = dict(
    type='TextxxxLocalVisualizer',  # 不同任务使用不同的可视化器
    vis_backends=vis_backends,
    name='visualizer')
```

依据以上示例，我们可以看出 Visualizer 的配置主要由两个部分组成，即，Visualizer 的类型以及其采用的可视化后端 vis_backends。

- 针对不同的 OCR 任务，MMOCR 中预置了多种可视化器件，包括 TextDetLocalVisualizer，TextRecogLocalVisualizer，TextSpottingLocalVisualizer 以及 KIELocalVisualizer。这些可视化器件依照自身任务的特点对基础的 Visulizer API 进行了拓展，并实现了相应的标签信息接口 add_datasamples。例如，用户可以直接使用 TextDetLocalVisualizer 来可视化文本检测任务的标签或预测结果。
- MMOCR 默认将可视化后端 vis_backend 设置为本地可视化后端 LocalVisBackend，将所有可视化结果及其他训练信息保存在本地文件夹中。

8.2 存储

MMOCR 默认使用本地可视化后端 LocalVisBackend，VisualizerHook 和 LoggerHook 中存储的模型损失、学习率、模型评估精度以及可视化结果等信息将被默认保存至 {work_dir}/{config_name}/{time}/{vis_data} 文件夹。此外，MMOCR 也支持其它常用的可视化后端，如 TensorboardVisBackend 以及 WandbVisBackend 用户只需要将配置文件中的 vis_backends 类型修改为对应的可视化后端即可。例如，用户只需要在配置文件中插入以下代码块，即可将数据存储至 TensorBoard 以及 WandB 中。

```
_base_.Visualizer.vis_backends = [
    dict(type='LocalVisBackend'),
    dict(type='TensorboardVisBackend'),
    dict(type='WandbVisBackend'),]
```

8.3 绘制

8.3.1 绘制预测结果信息

MMOCR 主要利用*VisualizationHook*validation 和 test 的预测结果, 默认情况下 VisualizationHook 为关闭状态，默认配置如下：

```
visualization=dict( # 用户可视化 validation 和 test 的结果
    type='VisualizationHook',
    enable=False,
    interval=1,
    show=False,
    draw_gt=False,
    draw_pred=False)
```

下表为 VisualizationHook 支持的参数：

如果在训练或者测试过程中想开启 VisualizationHook 相关功能和配置，仅需修改配置即可，以 dbnet_resnet18_fpnc_1200e_icdar2015.py 为例，同时绘制标注和预测，并且将图像展示，配置可进行如下修改

```
visualization = _base_.default_hooks.visualization
visualization.update(
    dict(enable=True, show=True, draw_gt=True, draw_pred=True))
```

如果只想查看预测结果信息可以只让 draw_pred=True

```
visualization = _base_.default_hooks.visualization
visualization.update(
    dict(enable=True, show=True, draw_gt=False, draw_pred=True))
```

在 test.py 过程中进一步简化，提供了 --show 和 --show-dir 两个参数，无需修改配置即可视化测试过程中绘制标注和预测结果。

```
# 展示 test 结果
python tools/test.py configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py␣
↪dbnet_r18_fpnc_1200e_icdar2015/epoch_400.pth --show

# 指定预测结果的存储位置
python tools/test.py configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py␣
↪dbnet_r18_fpnc_1200e_icdar2015/epoch_400.pth --show-dir imgs/
```

CHAPTER 9

常用工具

9.1 分析工具

9.1.1 数据集可视化工具

MMOCR 提供了数据集可视化工具 `tools/analysis_tools/browse_datasets.py` 以辅助用户排查可能遇到的数据集相关的问题。用户只需要指定所使用的训练配置文件路径，该工具即可自动将经过数据流水线（data pipeline）处理过的图像及其对应的真实标签绘制出来。例如，以下命令演示了如何使用该工具对“DBNet_R50_icdar2015”模型使用的训练数据进行可视化操作：

```
# 示例：可视化 dbnet_r50dcn_v2_fpnc_1200e_icadr2015 使用的训练数据
python tools/analysis_tools/browse_dataset.py configs/textdet/dbnet/dbnet_r50dcnv2_
↪fpnc_1200e_icdar2015.py
```

效果如下图所示：

基于此工具，用户可以方便地验证自定义数据集的标注格式是否正确；也可以通过修改配置文件中的 `train_pipeline` 来验证不同的数据增强策略组合是否符合自己的预期。`browse_dataset.py` 的可选参数如下：

9.1.2 离线评测工具

对于已保存的预测结果，我们提供了离线评测脚本 `tools/analysis_tools/offline_eval.py`。例如，以下代码演示了如何使用该工具对“PSENet”模型的输出结果进行离线评估：

```
# 初次运行测试脚本时，用户可以通过指定 --save-preds 参数来保存模型的输出结果
python tools/test.py ${CONFIG_FILE} ${CHECKPOINT_FILE} --save-preds
# 示例：对 PSENet 进行测试
python tools/test.py configs/textdet/psenet/psenet_r50_fpnf_600e_icdar2015.py epoch_
↪600.pth --save-preds

# 之后即可使用已保存的输出文件进行离线评估
python tools/analysis_tool/offline_eval.py ${CONFIG_FILE} ${PRED_FILE}
# 示例：对已保存的 PSENet 结果进行离线评估
python tools/analysis_tools/offline_eval.py configs/textdet/psenet/psenet_r50_fpnf_
↪600e_icdar2015.py work_dirs/psenet_r50_fpnf_600e_icdar2015/epoch_600.pth_
↪predictions.pkl
```

`--save-preds` 默认将输出结果保存至 `work_dir/CONFIG_NAME/MODEL_NAME_predictions.pkl`

此外，基于此工具，用户也可以将其他算法库获取的预测结果转换成 MMOCR 支持的格式，从而使用 MMOCR 内置的评估指标来对其他算法库的模型进行评测。

CHAPTER 10

设计理念与特性

待更新

CHAPTER 11

数据流

待更新

CHAPTER 12

数据集

待更新

CHAPTER 13

结构

待更新

CHAPTER 14

模型

待更新

CHAPTER 15

数据变换

待更新

CHAPTER 16

可视化组件

待更新

CHAPTER 17

评估

待更新

CHAPTER 18

开发默认约定

待更新

CHAPTER 19

引擎

待更新

CHAPTER 20

概览

伴随着 OpenMMLab 2.0 的发布，MMOCR 1.0 本身也作出了许多突破性的改变，使得代码的冗余度降低，代码效率提高，整体设计上也变得更为一致。然而，这些改变使得完美的后向兼容不再可能。我们也深知在这样巨大的变动之下，老用户想第一时间适应新版本也绝非易事。因此，我们推出了详细的迁移指南，旨在让老用户们尽可能平滑地过渡到全新的框架，最终能享受到全新的 MMOCR 和整个 OpenMMLab 2.0 生态系统为生产力带来的巨大优势。

接下来，请根据你的实际需求，阅读需要的章节：

- 如果你需要把 0.x 版本中训练的模型直接迁移到 1.0 版本中使用，请阅读预训练模型迁移
- 如果你需要训练模型，请阅读数据集迁移 和数据增强迁移
- 如果你需要在 MMOCR 上进行开发，请阅读代码迁移 和 上游依赖库变更

CHAPTER 21

代码结构变动

MMOCR 为了兼顾文本检测、识别和关键信息提取等任务，在初版设计时存在许多欠缺考虑的地方。在本次 1.0 版本的升级中，MMOCR 同步提出了新的模型架构，旨在尽量与 OpenMMLab 整体的设计对齐，且在算法库内部达成结构上的统一。虽然本次升级并非完全后向兼容，但所有的变动都是有迹可循的。因此，我们在本章节总结出了开发者可能会关心的改动，供有需要的用户参考。

21.1 整体改动

MMOCR 0.x 存在着对模块功能边界定义不清晰的问题。在 MMOCR 1.0 中，我们重构了模型模块的设计，并定义了它们的模块边界。

- 考虑到方向差异过大，MMOCR 1.0 中取消了对命名实体识别的支持。
- 模型中计算损失（loss）的部分模块被抽象化为 Module Loss，转换原始标注为损失目标（loss target）的功能也被包括在内。另一个模块抽象 Postprocessor 则负责在预测时解码模型原始输出为对应任务的 `DataSample`。
- 所有模型的输入简化为包含图像原始特征的 `inputs` 和图片元信息的 `List[DataSample]`。输出格式也得到统一，训练时是包含 loss 的字典，测试时的输出为包含预测结果的对应任务的 `DataSample`。
- Module Loss 来源于 0.x 版本中实现与单个模型强相关的 `XXLoss` 类，它们在 1.0 中均被统一重命名为 `XXModuleLoss` 的形式（如 `DBLoss` 被重命名为 `DBModuleLoss`），`head` 传入的 loss 配置参数名也从 `loss` 改为 `module_loss`。
- 与模型实现无关的通用损失类名称保持 `XXLoss` 的形式，并放置于 `mmocr/models/common/losses` 下，如*MaskedBCELoss*。

- mmocr/models/common/losses 下的改动：0.x 中 DiceLoss 被重名为*MaskedDiceLoss*。FocalLoss 被移除。
- 增加了起源于 label converter 的 Dictionary 模块，它会在文本识别和关键信息提取任务中被用到。

21.2 文本检测

21.2.1 关键改动（太长不看版）

- 旧版的模型权重仍然适用于新版，但需要将权重字典 state_dict 中以 bbox_head 开头的字段重命名为 det_head。
- 计算 target 有关的变换 XXTargets 被转移到了 XXModuleLoss 中。

21.2.2 SingleStageTextDetector

- 原本继承链为 mmdet.BaseDetector->SingleStageDetector->SingleStageTextDetector，现在改为直接继承自 BaseDetector，中间的 SingleStageDetector 被删除。
- bbox_head 改名为 det_head。
- train_cfg、test_cfg 和 pretrained 字段被移除。
- forward_train() 与 simple_test() 分别被重构为 loss() 与 predict() 方法。其中 simple_test() 中负责将模型原始输出拆分并输入 head.get_bounary() 的部分被整合进了 BaseTextDetPostProcessor 中。
- TextDetectorMixin 中只实现了 show_result() 方法，实现与 TextDetLocalVisualizer 重合，因此已经被移除。

21.2.3 Head

- HeadMixin 为 XXXHead 在 0.x 版本中必须继承的基类，现在被 BaseTextDetHead 代替。里面的 get_boundary() 和 resize_boundary() 方法被重写为 BaseTextDetPostProcessor 的 __call__() 和 rescale() 方法。

21.2.4 ModuleLoss

- 文本检测中特有的数据变换 `XXXTargets` 全部移动到 `XXXModuleLoss._get_target_single` 中，与生成 target 相关的配置不再在数据流水线（pipeline）中设置，转而在 `XXXLoss` 中被配置。例如，`DBNetTargets` 的实现被移动到 `DBModuleLoss._get_target_single()` 中，而用户可以通过设置 `DBModuleLoss` 的初始化参数来控制损失目标的生成。

21.2.5 Postprocessor

- 原本的 `XXXPostprocessor.__call__()` 中的逻辑转移到重构后的 `XXXPostprocessor.get_text_instances()`。
- `BasePostprocessor` 重构为 `BaseTextDetPostProcessor`，此基类会将模型输出的预测结果拆分并逐个进行处理，并支持根据 `scale_factor` 自动缩放输出的多边形（polygon）或界定框（bounding box）。

21.3 文本识别

21.3.1 关键改动（太长不看版）

- 由于字典序发生了变化，且存在部分模型架构上的 bug 被修复，旧版的识别模型权重已经不再能直接应用于 1.0 中，我们将会在后续为有需要的用户推出迁移脚本教程。
- 0.x 版本中的 SegOCR 支持暂时移除，TPS-CRNN 会在后续版本中被支持。
- 测试时增强（test time augmentation）在此版本中暂未支持，但将会在后续版本中更新。
- Label converter 模块被移除，里面的功能被拆分至 Dictionary, ModuleLoss 和 Postprocessor 模块中。
- 统一模型中对 `max_seq_len` 的定义为模型的原始输出长度。

21.3.2 Label Converter

- 原有的 label converter 存在拼写错误 (label convertor)，我们通过删除掉这个类规避了这个问题。
- 负责对字符/字符串与数字索引互相转换的部分被提取至 `Dictionary` 类中。
- 在旧版本中，不同的 label converter 会有不一样的特殊字符集和字符序。在 0.x 版本中，字符序如下：

在 1.0 中，我们不再以任务为边界设计不同的字典和字符序，取而代之的是统一了字符序的 Dictionary，其字符序为 characters, <BOS/EOS>, <PAD>, <UKN>。`CTCConvertor` 中 <BLK> 被等价替换为 <PAD>。

- `label_convertor` 中原本支持三种方式初始化字典：`dict_type`、`dict_file` 和 `dict_list`，现在在 `Dictionary` 中被简化为 `dict_file` 一种。同时，我们也把原本在 `dict_type` 中支持的字典格式转化为现在 `dicts/` 目录下的预设字典文件。对应映射如下：

- label_converter 中 str2tensor() 的实现被转移到 ModuleLoss.get_targets() 中。下面的表格列出了旧版与新版方法实现的对应关系。注意，新旧版的实现并非完全一致。
- label_converter 中 tensor2idx() 的实现被转移到 Postprocessor.get_single_prediction() 中。下面的表格列出了旧版与新版方法实现的对应关系。注意，新旧版的实现并非完全一致。

21.4 关键信息提取

21.4.1 关键改动（太长不看版）

- 由于模型的输入发生了变化，旧版模型的权重已经不再能直接应用于 1.0 中。

21.4.2 KIEDataset & OpensetKIEDataset

- 读取数据的部分被简化到 WildReceiptDataset 中。
- 对节点和边作额外处理的部分被转移到了 LoadKIEAnnotation 中。
- 使用字典对文本进行转化的部分被转移到了 SDMGRHead.convert_text() 中，使用 Dictionary 实现。
- 计算文本框之间关系的部分 compute_relation() 被转移到 SDMGRHead.compute_relations() 中，在模型内进行。
- 评估模型表现的部分被简化为 F1Metric。
- OpensetKIEDataset 中处理模型边输出的部分被整理到 SDMGRPostProcessor 中。

21.4.3 SDMGR

- show_result() 被整合到 KIEVisualizer 中。
- forward_test() 中对输出进行后处理的部分被整理到 SDMGRPostProcessor 中。

21.5 Utils 变动

原本散布在各处的功能函数现已被统一归类在 mmocr/utils/ 下。以下为该目录下各文件的作用域：

- bbox_utils.py：四边界定框（bounding box）有关的功能函数。
- check_argument.py：检查参数类型的功能函数。
- collect_env.py：收集运行环境的功能函数。
- data_converter_utils.py：用于数据集转换的功能函数。

- fileio.py：输入/输出有关的功能函数。
- img_utils.py：处理图片的功能函数。
- mask_utils.py：与掩码有关的功能函数。
- ocr.py：用于 MMOCR 推理的功能函数。
- parsers.py：解码文件的功能函数。
- polygon_utils.py：多边形的功能函数。
- setup_env.py：存放初始化 MMOCR 的功能函数。
- string_utils.py：存放字符串的功能函数。
- typing.py：存放 MMOCR 中常用数据类型的缩写。

CHAPTER 22

数据集迁移

在 OpenMMLab 2.0 系列算法库基于 MMEngine 设计了统一的数据集基类 BaseDataset，并制定了数据集标注文件规范。基于此，我们在 MMOCR 1.0 版本中重构了 OCR 任务数据集基类*OCRDataset*。以下文档将介绍 MMOCR 中新旧数据集格式的区别，以及如何将旧数据集迁移至新版本中。对于暂不方便进行数据迁移的用户，我们也在第三节提供了临时的代码兼容方案。

注解： 关键信息抽取任务仍采用原有的 WildReceipt 数据集标注格式。

22.1 旧版数据格式回顾

针对不同任务，MMOCR 0.x 版本实现了多种不同的数据集类型，如文本检测任务的 `IcdarDataset`，`TextDetDataset`；文本识别任务的 `OCRDataset`，`OCRSegDataset` 等。而不同的数据集类型同时还可能存在多种不同的标注及文件存储后端，如 `.txt`、`.json`、`.jsonl` 等，使得用户在自定义数据集时需要配置各类数据加载器 (`Loader`) 以及数据解析器 (`Parser`)。这不仅增加了用户的使用难度，也带来了许多问题和隐患。例如，以 `.txt` 格式存储的简单 `OCDDataset` 在遇到包含空格的文本标注时将会报错。

22.1.1 文本检测

文本检测任务中，`IcdarDataset` 采用了与通用目标检测 COCO 数据集一致的标注格式。

```
{
  "images": [
    {
      "id": 1,
      "width": 800,
      "height": 600,
      "file_name": "test.jpg"
    }
  ],
  "annotations": [
    {
      "id": 1,
      "image_id": 1,
      "category_id": 1,
      "bbox": [0,0,10,10],
      "segmentation": [
          [0,0,10,0,10,10,0,10]
      ],
      "area": 100,
      "iscrowd": 0
    }
  ]
}
```

而 `TextDetDataset` 则采用了 JSON Line 的存储格式，将类似 COCO 格式的标签转换成文本存放在 `.txt` 或 `.jsonl` 格式文件中。

```
{"file_name": "test/img_2.jpg", "height": 720, "width": 1280,  "annotations": [{
↪"iscrowd": 0, "category_id": 1, "bbox": [602.0, 173.0,  33.0, 24.0], "segmentation
↪": [[602, 173, 635, 175, 634, 197, 602,  196]]}, {"iscrowd": 0, "category_id": 1,
↪"bbox": [734.0, 310.0, 58.0,  54.0], "segmentation": [[734, 310, 792, 320, 792, 364,
↪ 738, 361]]}]}
{"file_name": "test/img_5.jpg", "height": 720, "width": 1280,  "annotations": [{
↪"iscrowd": 1, "category_id": 1, "bbox": [405.0, 409.0,  32.0, 52.0], "segmentation
↪": [[408, 409, 437, 436, 434, 461, 405,  433]]}, {"iscrowd": 1, "category_id": 1,
↪"bbox": [435.0, 434.0, 8.0,  33.0], "segmentation": [[437, 434, 443, 440, 441, 467,␣
↪435, 462]]}]}
```

22.1.2 文本识别

对于文本识别任务，MMOCR 0.x 版本中存在两种数据标注格式。其中 `.txt` 格式的标注文件每一行共有两个字段，分别存放了图片名以及标注的文本内容，并以空格分隔。

```
img1.jpg OpenMMLab
img2.jpg MMOCR
```

而 JSON Line 格式则使用 `json.dumps` 将 JSON 格式的标注转换为文本内容后存放在.jsonl 文件中，其内容形似一个字典，将文件名和文本标注信息分别存放在 `filename` 和 `text` 字段中。

```
{"filename": "img1.jpg", "text": "OpenMMLab"}
{"filename": "img2.jpg", "text": "MMOCR"}
```

22.2 新版数据格式

为解决 0.x 版本中数据集格式过于混杂的情况，MMOCR 1.x 采用了基于 MMEngine 设计的统一数据标准。每一个数据标注文件存放在 `.json` 文件中，并使用类似字典的格式分别存放了数据集的元信息（`metainfo`）与具体的标注内容（`data_list`）。

```
{
  "metainfo":
    {
      "classes": ("cat", "dog"),
      // ...
    },
  "data_list":
    [
      {
        "img_path": "xxx/xxx_0.jpg",
        "img_label": 0,
        // ...
      },
      // ...
    ]
}
```

基于此，我们针对 MMOCR 特有的任务设计了 `TextDetDataset`、`TextRecogDataset`。

22.2.1 文本检测

新版格式介绍

TextDetDataset 中存放了文本检测任务所需的边界盒标注、文件名等信息。由于文本检测任务中只有 1 个类别，因此我们将其类别 id 默认设置为 0，而背景类则为 1。tests/data/det_toy_dataset/instances_test.json 中存放了一个文本检测任务的数据标注示例，用户可以参考该文件来将自己的数据集转换为我们支持的格式。

```
{
  "metainfo":
    {
      "dataset_type": "TextDetDataset",
      "task_name": "textdet",
      "category": [{"id": 0, "name": "text"}]
    },
  "data_list":
    [
      {
        "img_path": "test_img.jpg",
        "height": 640,
        "width": 640,
        "instances":
          [
            {
              "polygon": [0, 0, 0, 10, 10, 20, 20, 0],
              "bbox": [0, 0, 10, 20],
              "bbox_label": 0,
              "ignore": False
            },
            // ...
          ]
      }
    ]
}
```

迁移脚本

为帮助用户将旧版本标注文件迁移至新格式，我们提供了迁移脚本。使用方法如下：

```
python tools/dataset_converters/textdet/data_migrator.py ${IN_PATH} ${OUT_PATH}
```

22.2.2 文本识别

新版格式介绍

TextRecogDataset 中存放了文本识别任务所需的文本内容，通常而言，文本识别数据集中的每一张图片都仅包含一个文本实例。我们在 tests/data/rec_toy_dataset/labels.json 提供了一个简单的识别数据格式示例，用户可以参考该文件以进一步了解其中的细节。

```
{
  "metainfo":
    {
      "dataset_type": "TextRecogDataset",
      "task_name": "textrecog",
    },
    "data_list":
    [
      {
        "img_path": "test_img.jpg",
        "instances":
            [
              {
                "text": "GRAND"
              }
            ]
      }
    ]
}
```

迁移脚本

为帮助用户将旧版本标注文件迁移至新格式，我们提供了迁移脚本。使用方法如下：

```
python tools/dataset_converters/textrecog/data_migrator.py ${IN_PATH} ${OUT_PATH} --
↪format ${txt, jsonl, lmdb}
```

22.3 兼容性

考虑到用户对数据迁移所需的成本，我们在 MMOCR 1.x 版本中暂时对 MMOCR 0.x 旧版本格式进行了兼容。

注解： 用于兼容旧数据格式的代码和组件可能在未来的版本中被完全移除。因此，我们强烈建议用户将数据集迁移至新的数据格式标准。

具体而言，我们提供了三个临时的数据集类*IcdarDataset*, *RecogTextDataset*, *RecogLMDBDataset* 来兼容旧格式的标注文件。分别对应了 MMOCR 0.x 版本中的文本检测数据集 `IcdarDataset`，`.txt`、`.jsonl` 和 `LMDB` 格式的文本识别数据标注。其使用方式与 0.x 版本一致。

1. *IcdarDataset* 支持 0.x 版本文本检测任务的 COCO 标注格式。只需要在 `configs/textdet/_base_/datasets` 中添加新的数据集配置文件，并指定其数据集类型为 `IcdarDataset` 即可。

```
data_root = 'data/det/icdar2015'

train_dataset = dict(
    type='IcdarDataset',
    data_root=data_root,
    ann_file='instances_training.json',
    data_prefix=dict(img_path='imgs/'),
    filter_cfg=dict(filter_empty_gt=True, min_size=32),
    pipeline=None)
```

2. *RecogTextDataset* 支持 0.x 版本文本识别任务的 `txt` 和 `jsonl` 标注格式。只需要在 `configs/textrecog/_base_/datasets` 中添加新的数据集配置文件，并指定其数据集类型为 `RecogTextDataset` 即可。例如，以下示例展示了如何配置并读取 toy dataset 中的旧格式标签 `old_label.txt` 以及 `old_label.jsonl`。

```
data_root = 'tests/data/rec_toy_dataset/'

# 读取旧版 txt 格式识别数据标签
txt_dataset = dict(
    type='RecogTextDataset',
    data_root=data_root,
    ann_file='old_label.txt',
    data_prefix=dict(img_path='imgs'),
    parser_cfg=dict(
        type='LineStrParser',
        keys=['filename', 'text'],
        keys_idx=[0, 1]),
    pipeline=[])
```

(下页继续)

(续上页)

```
# 读取旧版 json line 格式识别数据标签
jsonl_dataset = dict(
    type='RecogTextDataset',
    data_root=data_root,
    ann_file='old_label.jsonl',
    data_prefix=dict(img_path='imgs'),
    parser_cfg=dict(
        type='LineJsonParser',
        keys=['filename', 'text'],
    pipeline=[])
```

3. *RecogLMDBDataset* 支持 0.x 版本文本识别任务的 LMDB 标注格式。只需要在 configs/textrecog/_base_/datasets 中添加新的数据集配置文件，并指定其数据集类型为 RecogLMDBDataset 即可。例如，以下示例展示了如何配置并读取 toy dataset 中的 label.lmdb，该 lmdb 文件**仅包含标签信息**。

```
data_root = 'tests/data/rec_toy_dataset/'

lmdb_dataset = dict(
    type='RecogLMDBDataset',
    data_root=data_root,
    ann_file='label.lmdb',
    data_prefix=dict(img_path='imgs'),
    pipeline=[])
```

当 lmdb 文件中既包含标签信息又包含图像时，我们除了需要将数据集类型设定为 RecogLMDBDataset 以外，还需要将数据流水线中的图像读取方法由*LoadImageFromFile* 替换为*LoadImageFromLMDB*。

```
# 将数据集类型设定为 RecogLMDBDataset
data_root = 'tests/data/rec_toy_dataset/'

lmdb_dataset = dict(
    type='RecogLMDBDataset',
    data_root=data_root,
    ann_file='imgs.lmdb',
    data_prefix=dict(img_path='imgs.lmdb'), # 将 img_path 设定为 lmdb 文件名
    pipeline=[])
```

还需把 train_pipeline 及 test_pipeline 中的数据读取方法进行替换：

```
train_pipeline = [dict(type='LoadImageFromLMDB', color_type='grayscale', ignore_
↪empty=True)]
```

CHAPTER 23

预训练模型迁移指南

由于在新版本中我们对模型的结构进行了大量的重构和修复，MMOCR 1.x 并不能直接读入旧版的预训练权重。我们在网站上同步更新了所有模型的预训练权重和 log，供有需要的用户使用。

此外，我们正在进行针对文本检测任务的权重迁移工具的开发，并计划于近期版本内发布。由于文本识别和关键信息提取模型改动过大，且迁移是有损的，我们暂时不计划作相应支持。如果您有具体的需求，欢迎通过 Issue 向我们提问。

CHAPTER 24

数据变换迁移

24.1 简介

MMOCR 0.x 版本中，我们在 mmocr/datasets/pipelines/xxx_transforms.py 中实现了一系列的数据变换（Data Transforms）方法。然而，这些模块分散在各处，且缺乏规范统一的设计。因此，我们在 MMOCR 1.x 版本中对所有的数据增强模块进行了重构，并依照任务类型分别存放在 mmocr/datasets/transforms 目录下的 ocr_transforms.py，textdet_transforms.py 及 textrecog_transforms.py 中。其中，ocr_transforms.py 中实现了 OCR 相关任务通用的数据增强模块，而 textdet_transforms.py 和 textrecog_transforms.py 则分别实现了文本检测任务与文本识别任务相关的数据增强模组。

由于在重构过程中我们对部分模块进行了重命名、合并或拆分，使得新的调用接口与默认参数可能与旧版本存在不一致。因此，本文档将详细介绍如何对数据增强模块进行迁移，即，如何配置现有的数据变换来达到与旧版一致的行为。

24.2 配置迁移指南

24.2.1 数据格式化相关数据变换

1. Collect + CustomFormatBundle -> *PackTextDetInputs*/*PackTextRecogInputs*

PackxxxInputs 同时囊括了 Collect 和 CustomFormatBundle 两个功能，且不再有 key 参数，而训练目标 target 的生成现在被转移至在 loss 中完成。

```
dict(
    type='CustomFormatBundle',
    keys=['gt_shrink', 'gt_shrink_mask', 'gt_thr', 'gt_thr_mask'],
    meta_keys=['img_path', 'ori_shape', 'img_shape'],
    visualize=dict(flag=False, boundary_key='gt_shrink')),
dict(
    type='Collect',
    keys=['img', 'gt_shrink', 'gt_shrink_mask', 'gt_thr', 'gt_thr_mask'])
```

```
dict(
  type='PackTextDetInputs',
  meta_keys=('img_path', 'ori_shape', 'img_shape'))
```

24.2.2 数据增强相关数据变换

1. ResizeOCR -> *Resize*, *RescaleToHeight*, *PadToWidth*

   原有的 ResizeOCR 现在被拆分为三个独立的数据增强模块。

   keep_aspect_ratio=False 时，等价为 1.x 版本中的 Resize，其配置可按如下方式修改。

```
dict(
  type='ResizeOCR',
  height=32,
  min_width=100,
  max_width=100,
  keep_aspect_ratio=False)
```

```
dict(
  type='Resize',
  scale=(100, 32),
  keep_ratio=False)
```

keep_aspect_ratio=True，且 max_width=None 时。将图片的高缩放至固定值，并等比例缩放图像的宽。

```
dict(
  type='ResizeOCR',
  height=32,
  min_width=32,
  max_width=None,
  width_downsample_ratio = 1.0 / 16
  keep_aspect_ratio=True)
```

```
dict(
  type='RescaleToHeight',
  height=32,
  min_width=32,
  max_width=None,
  width_divisor=16),
```

keep_aspect_ratio=True，且 max_width 为固定值时。将图片的高缩放至固定值，并等比例缩放图像的宽。若缩放后的图像宽小于 max_width, 则将其填充至 max_width, 反之则将其裁剪至 max_width。即，输出图像的尺寸固定为 (height, max_width)。

```
dict(
  type='ResizeOCR',
  height=32,
  min_width=32,
  max_width=100,
  width_downsample_ratio = 1.0 / 16,
  keep_aspect_ratio=True)
```

```
dict(
  type='RescaleToHeight',
  height=32,
  min_width=32,
  max_width=100,
  width_divisor=16),
dict(
  type='PadToWidth',
  width=100)
```

2. RandomRotateTextDet & RandomRotatePolyInstances -> *RandomRotate*

   随机旋转数据增强策略已被整合至 RanomRotate。该方法的默认行为与 0.x 版本中的 RandomRotateTextDet 保持一致。此时仅需指定最大旋转角度 max_angle 即可。

注解： 新旧版本“max_angle”的默认值不同，因此需要重新进行指定。

```
dict(type='RandomRotateTextDet')
```

```
dict(type='RandomRotate', max_angle=10)
```

对于 RandomRotatePolyInstances，则需要指定参数 use_canvas=True。

```
dict(
  type='RandomRotatePolyInstances',
  rotate_ratio=0.5, # 指定概率为 0.5
  max_angle=60,
  pad_with_fixed_color=False)
```

```
# 用 RandomApply 对数据变换进行包装，并指定执行概率
dict(
  type='RandomApply',
  transforms=[
    dict(type='RandomRotate',
    max_angle=60,
    pad_with_fixed_color=False,
    use_canvas=True)],
  prob=0.5) # 设置执行概率为 0.5
```

注解： 在 0.x 版本中，部分数据增强方法通过定义一个内部变量 “xxx_ratio” 来指定执行概率，如 “rotate_ratio” ，“crop_ratio” 等。在 1.x 版本中，这些参数已被统一删除。现在，我们可以通过 “RandomApply” 来对不同的数据变换方法进行包装，并指定其执行概率。

---

3. RandomCropFlip -> *TextDetRandomCropFlip*

   目前仅对方法名进行了更改，其他参数保持一致。

4. RandomCropPolyInstances -> *RandomCrop*

   新版本移除了 crop_ratio 以及 instance_key，并统一使用 gt_polygons 为目标进行裁剪。

```
dict(
  type='RandomCropPolyInstances',
  instance_key='gt_masks',
  crop_ratio=0.8, # 指定概率为 0.8
  min_side_ratio=0.3)
```

```
# 用 RandomApply 对数据变换进行包装，并指定执行概率
dict(
  type='RandomApply',
  transforms=[dict(type='RandomCrop', min_side_ratio=0.3)],
  prob=0.8) # 设置执行概率为 0.8
```

5. RandomCropInstances -> *TextDetRandomCrop*

   新版本移除了 instance_key 和 mask_type，并统一使用 gt_polygons 为目标进行裁剪。

```
dict(
  type='RandomCropInstances',
  target_size=(800, 800),
  instance_key='gt_kernels')
```

```
dict(
  type='TextDetRandomCrop',
  target_size=(800, 800))
```

6. EastRandomCrop -> *RandomCrop* + *Resize* + mmengine.Pad

   原有的 EastRandomCrop 内同时对图像进行了剪裁、缩放以及填充。在新版本中，我们可以通过组合三种数据增强策略来达到相同的效果。

```
dict(
  type='EastRandomCrop',
  max_tries=10,
  min_crop_side_ratio=0.1,
  target_size=(640, 640))
```

```
dict(type='RandomCrop', min_side_ratio=0.1),
dict(type='Resize', scale=(640,640), keep_ratio=True),
dict(type='Pad', size=(640,640))
```

7. RandomScaling -> mmengine.RandomResize

   在新版本中，我们直接使用 MMEngine 中实现的 RandomResize 来代替原有的实现。

```
 dict(
  type='RandomScaling',
  size=800,
  scale=(0.75, 2.5))
```

```
dict(
  type='RandomResize',
  scale=(800, 800),
  ratio_range=(0.75, 2.5),
  keep_ratio=True)
```

注解： 默认地，数据流水线会从当前 *scope* 的注册器中搜索对应的数据变换，如果不存在该数据变换，则将继续在上游库，如 MMCV 及 MMEngine 中进行搜索。例如，MMOCR 中并未实现 RandomResize 方法，但我们仍然可以在配置中直接引用该数据增强方法，因为程序将自动从上游的 MMCV 中搜索该方法。此外，用户也可以通过添加前缀的形式来指定 *scope*。例如，mmengine.RandomResize 将强制指定使用 MMCV 库中实现的 RandomResize，当上下游库中存在同名方法时，则可以通过这种形式强制使用特定的版本。另外需要注意

的是，MMCV 中所有的数据变换方法都被注册至 MMEngine 中，因此我们使用 `mmengine.RandomResize` 而不是 `mmcv.RandomResize`。

8. `SquareResizePad` -> *`Resize` + `SourceImagePad`*

   原有的 `SquareResizePad` 内部实现了两个分支，并依据概率 `pad_ratio` 随机使用其中的一个分支进行数据增强。具体而言，一个分支先对图像缩放再填充；另一个分支则直接对图像进行缩放。为增强不同模块的复用性，我们在 1.x 版本中将该方法拆分成了 `Resize` + `SourceImagePad` 的组合形式，并通过 MMCV 中的 `RandomChoice` 来控制分支。

```
dict(
  type='SquareResizePad',
  target_size=800,
  pad_ratio=0.6)
```

```
dict(
  type='RandomChoice',
  transforms=[
    [
      dict(
        type='Resize',
        scale=800,
        keep_ratio=True),
      dict(
        type='SourceImagePad',
        target_scale=800)
    ],
    [
      dict(
        type='Resize',
        scale=800,
        keep_ratio=False)
    ]
  ],
  prob=[0.4, 0.6]), # 两种组合的选用概率
```

注解： 在 1.x 版本中，随机选择包装器“RandomChoice”代替了“OneOfWrapper”，可以从一系列数据变换组合中随机抽取一组并应用。

9. `RandomWrapper` -> `mmegnine.RandomApply`

   在 1.x 版本中，`RandomWrapper` 包装器被替换为由 MMCV 实现的 `RandomApply`，用以指定数据变换的执行概率。其中概率 `p` 现在被命名为 `prob`。

```
dict(
 type='RandomWrapper',
 p=0.25,
 transforms=[
     dict(type='PyramidRescale'),
 ])
```

```
dict(
  type='RandomApply',
  prob=0.25,
  transforms=[
    dict(type='PyramidRescale'),
  ])
```

10. `OneOfWrapper` -> `mmegnine.RandomChoice`

随机选择包装器现在被重命名为 `RandomChoice`，并且使用方法和原来完全一致。

11. `ScaleAspectJitter` -> *`ShortScaleAspectJitter`*, *`BoundedScaleAspectJitter`*

原有的 `ScaleAspectJitter` 实现了多种不同的图像尺寸抖动数据增强策略，在新版本中，我们将其拆分为数个逻辑更加清晰的独立数据变化方法。

`resize_type='indep_sample_in_range'` 时，其等价于图像在指定范围内的随机缩放。

```
dict(
  type='ScaleAspectJitter',
  img_scale=None,
  keep_ratio=False,
  resize_type='indep_sample_in_range',
  scale_range=(640, 2560))
```

```
 dict(
  type='RandomResize',
  scale=(640, 640),
  ratio_range=(1.0, 4.125),
  resize_type='Resize',
  keep_ratio=True))
```

`resize_type='long_short_bound'` 时，将图像缩放至指定大小，再对其长宽比进行抖动。这一逻辑现在由新的数据变换类 `BoundedScaleAspectJitter` 实现。

```
dict(
  type='ScaleAspectJitter',
  img_scale=[(3000, 736)],  # Unused
```

(下页继续)

(续上页)

```
  ratio_range=(0.7, 1.3),
  aspect_ratio_range=(0.9, 1.1),
  multiscale_mode='value',
  long_size_bound=800,
  short_size_bound=480,
  resize_type='long_short_bound',
  keep_ratio=False)
```

```
dict(
  type='BoundedScaleAspectJitter',
  long_size_bound=800,
  short_size_bound=480,
  ratio_range=(0.7, 1.3),
  aspect_ratio_range=(0.9, 1.1))
```

`resize_type='around_min_img_scale'`（默认参数）时，将图像的短边缩放至指定大小，再在指定范围内对长宽比进行抖动。最后，确保其边长能被 `scale_divisor` 整除。这一逻辑由新的数据变换类 `ShortScaleAspectJitter` 实现。

```
dict(
  type='ScaleAspectJitter',
  img_scale=[(3000, 640)],
  ratio_range=(0.7, 1.3),
  aspect_ratio_range=(0.9, 1.1),
  multiscale_mode='value',
  keep_ratio=False)
```

```
dict(
  type='ShortScaleAspectJitter',
  short_size=640,
  ratio_range=(0.7, 1.3),
  aspect_ratio_range=(0.9, 1.1),
  scale_divisor=32),
```

CHAPTER 25

文字检测

25.1 概览

文字检测任务的数据集应按如下目录配置：

```
├── ctw1500
|   ├── annotations
|   ├── imgs
|   ├── instances_test.json
|   └── instances_training.json
├── icdar2015
|   ├── imgs
|   ├── instances_test.json
|   └── instances_training.json
├── icdar2017
|   ├── imgs
|   ├── instances_training.json
|   └── instances_val.json
├── synthtext
|   ├── imgs
|   └── instances_training.lmdb
|       ├── data.mdb
|       └── lock.mdb
├── textocr
|   ├── train
```

(下页继续)

(续上页)

```
|   ├── instances_training.json
|   └── instances_val.json
├── totaltext
|   ├── imgs
|   ├── instances_test.json
|   └── instances_training.json
```

25.2 重要提醒

注解：若用户需要在 CTW1500, ICDAR 2015/2017 或 Totaltext 数据集上训练模型，请注意这些数据集中有部分图片的 EXIF 信息里保存着方向信息。MMCV 采用的 OpenCV 后端会默认根据方向信息对图片进行旋转；而由于数据集的标注是在原图片上进行的，这种冲突会使得部分训练样本失效。因此，用户应该在配置 pipeline 时使用 `dict(type='LoadImageFromFile', color_type='color_ignore_orientation')` 以避免 MMCV 的这一行为。（配置文件可参考 DBNet 的 pipeline 配置）

25.3 准备步骤

25.3.1 ICDAR 2015

- 第一步：从下载地址下载 `ch4_training_images.zip`、`ch4_test_images.zip`、`ch4_training_localization_transcription_gt.zip`、`Challenge4_Test_Task1_GT.zip` 四个文件，分别对应训练集数据、测试集数据、训练集标注、测试集标注。
- 第二步：运行以下命令，移动数据集到对应文件夹

```
mkdir icdar2015 && cd icdar2015
mkdir imgs && mkdir annotations
# 移动数据到目录:
mv ch4_training_images imgs/training
mv ch4_test_images imgs/test
# 移动标注到目录:
mv ch4_training_localization_transcription_gt annotations/training
mv Challenge4_Test_Task1_GT annotations/test
```

- 第三步：下载 instances_training.json 和 instances_test.json，并放入 `icdar2015` 文件夹里。或者也可以用以下命令直接生成 `instances_training.json` 和 `instances_test.json`:

```
python tools/data/textdet/icdar_converter.py /path/to/icdar2015 -o /path/to/icdar2015␣
↪-d icdar2015 --split-list training test
```

25.3.2 ICDAR 2017

- 与上述步骤类似。

25.3.3 CTW1500

- 第一步：执行以下命令，从 下载地址 下载 train_images.zip，test_images.zip，train_labels.zip，test_labels.zip 四个文件并配置到对应目录：

```
mkdir ctw1500 && cd ctw1500
mkdir imgs && mkdir annotations

# 下载并配置标注
cd annotations
wget -O train_labels.zip https://universityofadelaide.box.com/shared/static/
↪jikuazluzyj4lq6umzei7m2ppmt3afyw.zip
wget -O test_labels.zip https://cloudstor.aarnet.edu.au/plus/s/uoeFl0pCN9BOCN5/
↪download
unzip train_labels.zip && mv ctw1500_train_labels training
unzip test_labels.zip -d test
cd ..
# 下载并配置数据
cd imgs
wget -O train_images.zip https://universityofadelaide.box.com/shared/static/
↪py5uwlfyyytbb2pxzq9czvu6fuqbjdh8.zip
wget -O test_images.zip https://universityofadelaide.box.com/shared/static/
↪t4w48ofnqkdw7jyc4t11nsukoeqk9c3d.zip
unzip train_images.zip && mv train_images training
unzip test_images.zip && mv test_images test
```

- 第二步：执行以下命令，生成 instances_training.json 和 instances_test.json。

```
python tools/data/textdet/ctw1500_converter.py /path/to/ctw1500 -o /path/to/ctw1500 --
↪split-list training test
```

25.3.4 SynthText

- 下载 data.mdb 和 lock.mdb 并放置到 `synthtext/instances_training.lmdb/` 中.

25.3.5 TextOCR

- 第一步：下载 train_val_images.zip，TextOCR_0.1_train.json 和 TextOCR_0.1_val.json 到 `textocr` 文件夹里。

```
mkdir textocr && cd textocr

# 下载 TextOCR 数据集
wget https://dl.fbaipublicfiles.com/textvqa/images/train_val_images.zip
wget https://dl.fbaipublicfiles.com/textvqa/data/textocr/TextOCR_0.1_train.json
wget https://dl.fbaipublicfiles.com/textvqa/data/textocr/TextOCR_0.1_val.json

# 把图片移到对应目录
unzip -q train_val_images.zip
mv train_images train
```

- 第二步：生成 `instances_training.json` 和 `instances_val.json`:

```
python tools/data/textdet/textocr_converter.py /path/to/textocr
```

25.3.6 Totaltext

- 第一步：从 github dataset 下载 `totaltext.zip`，从 github Groundtruth 下载 `groundtruth_text.zip`。（建议下载 `.mat` 格式的标注文件，因为我们提供的标注格式转换脚本 `totaltext_converter.py` 仅支持 `.mat` 格式。）

```
mkdir totaltext && cd totaltext
mkdir imgs && mkdir annotations

# 图像
# 在 ./totaltext 中执行
unzip totaltext.zip
mv Images/Train imgs/training
mv Images/Test imgs/test

# 标注文件
unzip groundtruth_text.zip
cd Groundtruth
```

(下页继续)

(续上页)

```
mv Polygon/Train ../annotations/training
mv Polygon/Test ../annotations/test
```

- 第二步：用以下命令生成 instances_training.json 和 instances_test.json：

```
python tools/data/textdet/totaltext_converter.py /path/to/totaltext -o /path/to/
↪totaltext --split-list training test
```

CHAPTER 26

文字识别

警告： 该章节翻译落后于英文版文档。

26.1 概览

文字识别任务的数据集应按如下目录配置：

```
├── mixture
|   ├── coco_text
|   |   ├── train_label.txt
|   |   ├── train_words
|   ├── icdar_2011
|   |   ├── training_label.txt
|   |   ├── Challenge1_Training_Task3_Images_GT
|   ├── icdar_2013
|   |   ├── train_label.txt
|   |   ├── test_label_1015.txt
|   |   ├── test_label_1095.txt
|   |   ├── Challenge2_Training_Task3_Images_GT
|   |   ├── Challenge2_Test_Task3_Images
|   ├── icdar_2015
|   |   ├── train_label.txt
|   |   ├── test_label.txt
```

(下页继续)

(续上页)

```
|   |   ├── ch4_training_word_images_gt
|   |   ├── ch4_test_word_images_gt
|   ├── III5K
|   |   ├── train_label.txt
|   |   ├── test_label.txt
|   |   ├── train
|   |   ├── test
|   ├── ct80
|   |   ├── test_label.txt
|   |   ├── image
|   ├── svt
|   |   ├── test_label.txt
|   |   ├── image
|   ├── svtp
|   |   ├── test_label.txt
|   |   ├── image
|   ├── Syn90k
|   |   ├── shuffle_labels.txt
|   |   ├── label.txt
|   |   ├── label.lmdb
|   |   ├── mnt
|   ├── SynthText
|   |   ├── alphanumeric_labels.txt
|   |   ├── shuffle_labels.txt
|   |   ├── instances_train.txt
|   |   ├── label.txt
|   |   ├── label.lmdb
|   |   ├── synthtext
|   ├── SynthAdd
|   |   ├── label.txt
|   |   ├── label.lmdb
|   |   ├── SynthText_Add
|   ├── TextOCR
|   |   ├── image
|   |   ├── train_label.txt
|   |   ├── val_label.txt
|   ├── Totaltext
|   |   ├── imgs
|   |   ├── annotations
|   |   ├── train_label.txt
|   |   ├── test_label.txt
|   ├── OpenVINO
|   |   ├── image_1
```

(下页继续)

(续上页)

```
|   |   ├── image_2
|   |   ├── image_5
|   |   ├── image_f
|   |   ├── image_val
|   |   ├── train_1_label.txt
|   |   ├── train_2_label.txt
|   |   ├── train_5_label.txt
|   |   ├── train_f_label.txt
|   |   ├── val_label.txt
```

(*) 注：由于官方的下载地址已经无法访问，我们提供了一个非官方的地址以供参考，但我们无法保证数据的准确性。

26.2 准备步骤

26.2.1 ICDAR 2013

- 第一步：从 下载地址 下载 `Challenge2_Test_Task3_Images.zip` 和 `Challenge2_Training_Task3_Images_GT.zip`
- 第二步：下载 test_label_1015.txt 和 train_label.txt

26.2.2 ICDAR 2015

- 第一步：从 下载地址 下载 `ch4_training_word_images_gt.zip` 和 `ch4_test_word_images_gt.zip`
- 第二步：下载 train_label.txt and test_label.txt

26.2.3 IIIT5K

- 第一步：从 下载地址 下载 `IIIT5K-Word_V3.0.tar.gz`
- 第二步：下载 train_label.txt 和 test_label.txt

26.2.4 svt

- 第一步：从 下载地址 下载 `svt.zip`
- 第二步：下载 test_label.txt
- 第三步：

```
python tools/data/textrecog/svt_converter.py <download_svt_dir_path>
```

26.2.5 ct80

- 第一步：下载 test_label.txt

26.2.6 svtp

- 第一步：下载 test_label.txt

26.2.7 coco_text

- 第一步：从 下载地址 下载文件
- 第二步：下载 train_label.txt

26.2.8 MJSynth (Syn90k)

- 第一步：从 下载地址 下载 `mjsynth.tar.gz`
- 第二步：下载 shuffle_labels.txt
- 第三步：

```
mkdir Syn90k && cd Syn90k

mv /path/to/mjsynth.tar.gz .

tar -xzf mjsynth.tar.gz

mv /path/to/shuffle_labels.txt .
mv /path/to/label.txt .

# 创建软链接
cd /path/to/mmocr/data/mixture
```

(下页继续)

(续上页)

```
ln -s /path/to/Syn90k Syn90k
```

26.2.9 SynthText (Synth800k)

- 第一步：下载 `SynthText.zip`: 下载地址
- 第二步：请根据你的实际需要，从下列标注中选择最适合的下载：label.txt （7,266,686 个标注）；shuffle_labels.txt （2,400,000 个随机采样的标注）；alphanumeric_labels.txt （7,239,272 个仅包含数字和字母的标注）；instances_train.txt （7,266,686 个字符级别的标注）。
- 第三步：

```
mkdir SynthText && cd SynthText
mv /path/to/SynthText.zip .
unzip SynthText.zip
mv SynthText synthtext

mv /path/to/shuffle_labels.txt .
mv /path/to/label.txt .
mv /path/to/alphanumeric_labels.txt .
mv /path/to/instances_train.txt .

# 创建软链接
cd /path/to/mmocr/data/mixture
ln -s /path/to/SynthText SynthText
```

- 第四步：生成裁剪后的图像和标注：

```
cd /path/to/mmocr

python tools/data/textrecog/synthtext_converter.py data/mixture/SynthText/gt.mat data/
↪mixture/SynthText/ data/mixture/SynthText/synthtext/SynthText_patch_horizontal --n_
↪proc 8
```

26.2.10 SynthAdd

- 第一步：从 SynthAdd (code:627x) 下载 `SynthText_Add.zip`
- 第二步：下载 label.txt
- 第三步：

````
mkdir SynthAdd && cd SynthAdd

mv /path/to/SynthText_Add.zip .

unzip SynthText_Add.zip

mv /path/to/label.txt .

# 创建软链接
cd /path/to/mmocr/data/mixture

````{tip}
运行以下命令，可以把 `.txt` 格式的标注文件转换成 `.lmdb` 格式：
```bash
python tools/data/utils/txt2lmdb.py -i <txt_label_path> -o <lmdb_label_path>
````

例如：

```
python tools/data/utils/txt2lmdb.py -i data/mixture/Syn90k/label.txt -o data/mixture/
↪Syn90k/label.lmdb
```

````
### TextOCR
  - 第一步：下载 [train_val_images.zip](https://dl.fbaipublicfiles.com/textvqa/images/
↪train_val_images.zip)，[TextOCR_0.1_train.json](https://dl.fbaipublicfiles.com/
↪textvqa/data/textocr/TextOCR_0.1_train.json) 和 [TextOCR_0.1_val.json](https://dl.
↪fbaipublicfiles.com/textvqa/data/textocr/TextOCR_0.1_val.json) 到 `textocr/` 目录。
  ```bash
  mkdir textocr && cd textocr

  # 下载 TextOCR 数据集
  wget https://dl.fbaipublicfiles.com/textvqa/images/train_val_images.zip
  wget https://dl.fbaipublicfiles.com/textvqa/data/textocr/TextOCR_0.1_train.json
  wget https://dl.fbaipublicfiles.com/textvqa/data/textocr/TextOCR_0.1_val.json

  # 对于数据图像
  unzip -q train_val_images.zip
  mv train_images train
  ```
  - 第二步：用四个并行进程剪裁图像然后生成 `train_label.txt`，`val_label.txt` ，可以使用以下命
令：
  ```bash
  python tools/data/textrecog/textocr_converter.py /path/to/textocr 4
  ```
````

(下页继续)

(续上页)

````
### Totaltext
  - 第一步：从 [github dataset](https://github.com/cs-chan/Total-Text-Dataset/tree/
↪master/Dataset) 下载 `totaltext.zip`，然后从 [github Groundtruth](https://github.com/
↪cs-chan/Total-Text-Dataset/tree/master/Groundtruth/Text) 下载 `groundtruth_text.zip`
(我们建议下载 `.mat` 格式的标注文件，因为我们提供的 `totaltext_converter.py` 标注格式转换工具只支
持 `.mat` 文件)
  ```bash
  mkdir totaltext && cd totaltext
  mkdir imgs && mkdir annotations

  # 对于图像数据
  # 在 ./totaltext 目录下运行
  unzip totaltext.zip
  mv Images/Train imgs/training
  mv Images/Test imgs/test

  # 对于标注文件
  unzip groundtruth_text.zip
  cd Groundtruth
  mv Polygon/Train ../annotations/training
  mv Polygon/Test ../annotations/test
  ```
  - 第二步：用以下命令生成经剪裁后的标注文件 `train_label.txt` 和 `test_label.txt` (剪裁后的图
像会被保存在目录 `data/totaltext/dst_imgs/`):
  ```bash
  python tools/data/textrecog/totaltext_converter.py /path/to/totaltext -o /path/to/
↪totaltext --split-list training test
  ```

### OpenVINO
  - 第零步：安装 [awscli](https://aws.amazon.com/cli/)。
  - 第一步：下载 [Open Images](https://github.com/cvdfoundation/open-images-dataset
↪#download-images-with-bounding-boxes-annotations) 的子数据集 `train_1`、 `train_2`、␣
↪`train_5`、 `train_f` 及 `validation` 至 `openvino/`。
  ```bash
  mkdir openvino && cd openvino

  # 下载 Open Images 的子数据集
  for s in 1 2 5 f; do
    aws s3 --no-sign-request cp s3://open-images-dataset/tar/train_${s}.tar.gz .
  done
````

(下页继续)
````

(续上页)

```
  aws s3 --no-sign-request cp s3://open-images-dataset/tar/validation.tar.gz .

  # 下载标注文件
  for s in 1 2 5 f; do
    wget https://storage.openvinotoolkit.org/repositories/openvino_training_
↪extensions/datasets/open_images_v5_text/text_spotting_openimages_v5_train_${s}.json
  done
  wget https://storage.openvinotoolkit.org/repositories/openvino_training_extensions/
↪datasets/open_images_v5_text/text_spotting_openimages_v5_validation.json

  # 解压数据集
  mkdir -p openimages_v5/val
  for s in 1 2 5 f; do
    tar zxf train_${s}.tar.gz -C openimages_v5
  done
  tar zxf validation.tar.gz -C openimages_v5/val
  ```
  - 第二步： 运行以下的命令，以用 4 个进程生成标注 `train_{1,2,5,f}_label.txt` 和 `val_label.
↪txt` 并裁剪原图：
  ```bash
  python tools/data/textrecog/openvino_converter.py /path/to/openvino 4
  ```
```
```

CHAPTER 27

关键信息提取

27.1 概览

关键信息提取任务的数据集，文件目录应按如下配置：

```
└── wildreceipt
  ├── class_list.txt
  ├── dict.txt
  ├── image_files
  ├── test.txt
  └── train.txt
```

27.2 准备步骤

27.2.1 WildReceipt

- 下载并解压 wildreceipt.tar

27.2.2 WildReceiptOpenset

- 准备好 WildReceipt。
- 转换 WildReceipt 成 OpenSet 格式:

```
# 你可以运行以下命令以获取更多可用参数:
# python tools/data/kie/closeset_to_openset.py -h
python tools/data/kie/closeset_to_openset.py data/wildreceipt/train.txt data/
↪wildreceipt/openset_train.txt
python tools/data/kie/closeset_to_openset.py data/wildreceipt/test.txt data/
↪wildreceipt/openset_test.txt
```

注解: 这篇教程里讲述了更多 CloseSet 和 OpenSet 数据格式之间的区别。

CHAPTER 28

统计数据

- 模型权重文件数量：17
- 配置文件数量：25
- 论文数量：16
  - ALGORITHM: 16

28.1 关键信息提取模型

- 模型权重文件数量：3
- 配置文件数量：3
- 论文数量：1
  - [ALGORITHM] Spatial Dual-Modality Graph Reasoning for Key Information Extraction

28.2 文本检测模型

- 模型权重文件数量：13
- 配置文件数量：13
- 论文数量：8
  - [ALGORITHM] Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection
  - [ALGORITHM] Efficient and Accurate Arbitrary-Shaped Text Detection With Pixel Aggregation Network
  - [ALGORITHM] Fourier Contour Embedding for Arbitrary-Shaped Text Detection
  - [ALGORITHM] Mask R-CNN
  - [ALGORITHM] Real-Time Scene Text Detection With Differentiable Binarization and Adaptive Scale Fusion
  - [ALGORITHM] Real-Time Scene Text Detection With Differentiable Binarization
  - [ALGORITHM] Shape Robust Text Detection With Progressive Scale Expansion Network
  - [ALGORITHM] Textsnake: A Flexible Representation for Detecting Text of Arbitrary Shapes

28.3 文本识别模型

- 模型权重文件数量：1
- 配置文件数量：9
- 论文数量：7
  - [ALGORITHM] An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition
  - [ALGORITHM] Nrtr: A No-Recurrence Sequence-to-Sequence Model for Scene Text Recognition
  - [ALGORITHM] On Recognizing Texts of Arbitrary Shapes With 2d Self-Attention
  - [ALGORITHM] Read Like Humans: Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition
  - [ALGORITHM] Robustscanner: Dynamically Enhancing Positional Clues for Robust Text Recognition
  - [ALGORITHM] Show, Attend and Read: A Simple and Strong Baseline for Irregular Text Recognition
  - [ALGORITHM] {Master

CHAPTER 29

文本检测模型

29.1 DBNet

Real-time Scene Text Detection with Differentiable Binarization

29.1.1 Abstract

Recently, segmentation-based methods are quite popular in scene text detection, as the segmentation results can more accurately describe scene text of various shapes such as curve text. However, the post-processing of binarization is essential for segmentation-based detection, which converts probability maps produced by a segmentation method into bounding boxes/regions of text. In this paper, we propose a module named Differentiable Binarization (DB), which can perform the binarization process in a segmentation network. Optimized along with a DB module, a segmentation network can adaptively set the thresholds for binarization, which not only simplifies the post-processing but also enhances the performance of text detection. Based on a simple segmentation network, we validate the performance improvements of DB on five benchmark datasets, which consistently achieves state-of-the-art results, in terms of both detection accuracy and speed. In particular, with a light-weight backbone, the performance improvements by DB are significant so that we can look for an ideal tradeoff between detection accuracy and efficiency. Specifically, with a backbone of ResNet-18, our detector achieves an F-measure of 82.8, running at 62 FPS, on the MSRA-TD500 dataset.

29.1.2 Results and models

ICDAR2015

29.1.3 Citation

```
@article{Liao_Wan_Yao_Chen_Bai_2020,
    title={Real-Time Scene Text Detection with Differentiable Binarization},
    journal={Proceedings of the AAAI Conference on Artificial Intelligence},
    author={Liao, Minghui and Wan, Zhaoyi and Yao, Cong and Chen, Kai and Bai, Xiang},
    year={2020},
    pages={11474-11481}}
```

29.2 DBNetpp

Real-Time Scene Text Detection with Differentiable Binarization and Adaptive Scale Fusion

29.2.1 Abstract

Recently, segmentation-based scene text detection methods have drawn extensive attention in the scene text detection field, because of their superiority in detecting the text instances of arbitrary shapes and extreme aspect ratios, profiting from the pixel-level descriptions. However, the vast majority of the existing segmentation-based approaches are limited to their complex post-processing algorithms and the scale robustness of their segmentation models, where the post-processing algorithms are not only isolated to the model optimization but also time-consuming and the scale robustness is usually strengthened by fusing multi-scale feature maps directly. In this paper, we propose a Differentiable Binarization (DB) module that integrates the binarization process, one of the most important steps in the post-processing procedure, into a segmentation network. Optimized along with the proposed DB module, the segmentation network can produce more accurate results, which enhances the accuracy of text detection with a simple pipeline. Furthermore, an efficient Adaptive Scale Fusion (ASF) module is proposed to improve the scale robustness by fusing features of different scales adaptively. By incorporating the proposed DB and ASF with the segmentation network, our proposed scene text detector consistently achieves state-of-the-art results, in terms of both detection accuracy and speed, on five standard benchmarks.

29.2.2 Results and models

ICDAR2015

29.2.3 Citation

```
@article{liao2022real,
    title={Real-Time Scene Text Detection with Differentiable Binarization and
↪Adaptive Scale Fusion},
    author={Liao, Minghui and Zou, Zhisheng and Wan, Zhaoyi and Yao, Cong and Bai,
↪Xiang},
    journal={IEEE Transactions on Pattern Analysis and Machine Intelligence},
    year={2022},
    publisher={IEEE}
}
```

29.3 DRRG

Deep relational reasoning graph network for arbitrary shape text detection

29.3.1 Abstract

Arbitrary shape text detection is a challenging task due to the high variety and complexity of scenes texts. In this paper, we propose a novel unified relational reasoning graph network for arbitrary shape text detection. In our method, an innovative local graph bridges a text proposal model via Convolutional Neural Network (CNN) and a deep relational reasoning network via Graph Convolutional Network (GCN), making our network end-to-end trainable. To be concrete, every text instance will be divided into a series of small rectangular components, and the geometry attributes (e.g., height, width, and orientation) of the small components will be estimated by our text proposal model. Given the geometry attributes, the local graph construction model can roughly establish linkages between different text components. For further reasoning and deducing the likelihood of linkages between the component and its neighbors, we adopt a graph-based network to perform deep relational reasoning on local graphs. Experiments on public available datasets demonstrate the state-of-the-art performance of our method.

29.3.2 Results and models

CTW1500

29.3.3 Citation

```
@article{zhang2020drrg,
  title={Deep relational reasoning graph network for arbitrary shape text detection},
  author={Zhang, Shi-Xue and Zhu, Xiaobin and Hou, Jie-Bo and Liu, Chang and Yang,
↪Chun and Wang, Hongfa and Yin, Xu-Cheng},
  booktitle={CVPR},
  pages={9699-9708},
```

(下页继续)

(续上页)

```
  year={2020}
}
```

29.4 FCENet

Fourier Contour Embedding for Arbitrary-Shaped Text Detection

29.4.1 Abstract

One of the main challenges for arbitrary-shaped text detection is to design a good text instance representation that allows networks to learn diverse text geometry variances. Most of existing methods model text instances in image spatial domain via masks or contour point sequences in the Cartesian or the polar coordinate system. However, the mask representation might lead to expensive post-processing, while the point sequence one may have limited capability to model texts with highly-curved shapes. To tackle these problems, we model text instances in the Fourier domain and propose one novel Fourier Contour Embedding (FCE) method to represent arbitrary shaped text contours as compact signatures. We further construct FCENet with a backbone, feature pyramid networks (FPN) and a simple post-processing with the Inverse Fourier Transformation (IFT) and Non-Maximum Suppression (NMS). Different from previous methods, FCENet first predicts compact Fourier signatures of text instances, and then reconstructs text contours via IFT and NMS during test. Extensive experiments demonstrate that FCE is accurate and robust to fit contours of scene texts even with highly-curved shapes, and also validate the effectiveness and the good generalization of FCENet for arbitrary-shaped text detection. Furthermore, experimental results show that our FCENet is superior to the state-of-the-art (SOTA) methods on CTW1500 and Total-Text, especially on challenging highly-curved text subset.

29.4.2 Results and models

CTW1500

ICDAR2015

29.4.3 Citation

```
@InProceedings{zhu2021fourier,
      title={Fourier Contour Embedding for Arbitrary-Shaped Text Detection},
      author={Yiqin Zhu and Jianyong Chen and Lingyu Liang and Zhanghui Kuang and␣
↪Lianwen Jin and Wayne Zhang},
      year={2021},
      booktitle = {CVPR}
      }
```

29.5 Mask R-CNN

Mask R-CNN

29.5.1 Abstract

We present a conceptually simple, flexible, and general framework for object instance segmentation. Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover, Mask R-CNN is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework. We show top results in all three tracks of the COCO suite of challenges, including instance segmentation, bounding-box object detection, and person keypoint detection. Without bells and whistles, Mask R-CNN outperforms all existing, single-model entries on every task, including the COCO 2016 challenge winners. We hope our simple and effective approach will serve as a solid baseline and help ease future research in instance-level recognition.

29.5.2 Results and models

CTW1500

ICDAR2015

29.5.3 Citation

```
@INPROCEEDINGS{8237584,
  author={K. {He} and G. {Gkioxari} and P. {Dollár} and R. {Girshick}},
  booktitle={2017 IEEE International Conference on Computer Vision (ICCV)},
  title={Mask R-CNN},
  year={2017},
  pages={2980-2988},
  doi={10.1109/ICCV.2017.322}}
```

29.6 PANet

Efficient and Accurate Arbitrary-Shaped Text Detection with Pixel Aggregation Network

29.6.1 Abstract

Scene text detection, an important step of scene text reading systems, has witnessed rapid development with convolutional neural networks. Nonetheless, two main challenges still exist and hamper its deployment to real-world applications. The first problem is the trade-off between speed and accuracy. The second one is to model the arbitrary-shaped text instance. Recently, some methods have been proposed to tackle arbitrary-shaped text detection, but they rarely take the speed of the entire pipeline into consideration, which may fall short in practical this http URL this paper, we propose an efficient and accurate arbitrary-shaped text detector, termed Pixel Aggregation Network (PAN), which is equipped with a low computational-cost segmentation head and a learnable post-processing. More specifically, the segmentation head is made up of Feature Pyramid Enhancement Module (FPEM) and Feature Fusion Module (FFM). FPEM is a cascadable U-shaped module, which can introduce multi-level information to guide the better segmentation. FFM can gather the features given by the FPEMs of different depths into a final feature for segmentation. The learnable post-processing is implemented by Pixel Aggregation (PA), which can precisely aggregate text pixels by predicted similarity vectors. Experiments on several standard benchmarks validate the superiority of the proposed PAN. It is worth noting that our method can achieve a competitive F-measure of 79.9% at 84.2 FPS on CTW1500.

29.6.2 Results and models

CTW1500

ICDAR2015

29.6.3 Citation

```
@inproceedings{WangXSZWLYS19,
  author={Wenhai Wang and Enze Xie and Xiaoge Song and Yuhang Zang and Wenjia Wang␣
↪and Tong Lu and Gang Yu and Chunhua Shen},
  title={Efficient and Accurate Arbitrary-Shaped Text Detection With Pixel␣
↪Aggregation Network},
  booktitle={ICCV},
  pages={8439--8448},
  year={2019}
  }
```

29.7 PSENet

Shape robust text detection with progressive scale expansion network

29.7.1 Abstract

Scene text detection has witnessed rapid progress especially with the recent development of convolutional neural networks. However, there still exists two challenges which prevent the algorithm into industry applications. On the one hand, most of the state-of-art algorithms require quadrangle bounding box which is in-accurate to locate the texts with arbitrary shape. On the other hand, two text instances which are close to each other may lead to a false detection which covers both instances. Traditionally, the segmentation-based approach can relieve the first problem but usually fail to solve the second challenge. To address these two challenges, in this paper, we propose a novel Progressive Scale Expansion Network (PSENet), which can precisely detect text instances with arbitrary shapes. More specifically, PSENet generates the different scale of kernels for each text instance, and gradually expands the minimal scale kernel to the text instance with the complete shape. Due to the fact that there are large geometrical margins among the minimal scale kernels, our method is effective to split the close text instances, making it easier to use segmentation-based methods to detect arbitrary-shaped text instances. Extensive experiments on CTW1500, Total-Text, ICDAR 2015 and ICDAR 2017 MLT validate the effectiveness of PSENet. Notably, on CTW1500, a dataset full of long curve texts, PSENet achieves a F-measure of 74.3% at 27 FPS, and our best F-measure (82.2%) outperforms state-of-art algorithms by 6.6%. The code will be released in the future.

29.7.2 Results and models

CTW1500

ICDAR2015

29.7.3 Citation

```
@inproceedings{wang2019shape,
  title={Shape robust text detection with progressive scale expansion network},
  author={Wang, Wenhai and Xie, Enze and Li, Xiang and Hou, Wenbo and Lu, Tong and Yu,
↪ Gang and Shao, Shuai},
  booktitle={Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern␣
↪Recognition},
  pages={9336--9345},
  year={2019}
}
```

29.8 Textsnake

TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes

29.8.1 Abstract

Driven by deep neural networks and large scale datasets, scene text detection methods have progressed substantially over the past years, continuously refreshing the performance records on various standard benchmarks. However, limited by the representations (axis-aligned rectangles, rotated rectangles or quadrangles) adopted to describe text, existing methods may fall short when dealing with much more free-form text instances, such as curved text, which are actually very common in real-world scenarios. To tackle this problem, we propose a more flexible representation for scene text, termed as TextSnake, which is able to effectively represent text instances in horizontal, oriented and curved forms. In TextSnake, a text instance is described as a sequence of ordered, overlapping disks centered at symmetric axes, each of which is associated with potentially variable radius and orientation. Such geometry attributes are estimated via a Fully Convolutional Network (FCN) model. In experiments, the text detector based on TextSnake achieves state-of-the-art or comparable performance on Total-Text and SCUT-CTW1500, the two newly published benchmarks with special emphasis on curved text in natural images, as well as the widely-used datasets ICDAR 2015 and MSRA-TD500. Specifically, TextSnake outperforms the baseline on Total-Text by more than 40% in F-measure.

29.8.2 Results and models

CTW1500

29.8.3 Citation

```
@article{long2018textsnake,
  title={TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes},
  author={Long, Shangbang and Ruan, Jiaqiang and Zhang, Wenjie and He, Xin and Wu,␣
↪Wenhao and Yao, Cong},
  booktitle={ECCV},
  pages={20-36},
  year={2018}
}
```

CHAPTER 30

文本识别模型

30.1 ABINet

Read Like Humans: Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition

30.1.1 Abstract

Linguistic knowledge is of great benefit to scene text recognition. However, how to effectively model linguistic rules in end-to-end deep networks remains a research challenge. In this paper, we argue that the limited capacity of language models comes from: 1) implicitly language modeling; 2) unidirectional feature representation; and 3) language model with noise input. Correspondingly, we propose an autonomous, bidirectional and iterative ABINet for scene text recognition. Firstly, the autonomous suggests to block gradient flow between vision and language models to enforce explicitly language modeling. Secondly, a novel bidirectional cloze network (BCN) as the language model is proposed based on bidirectional feature representation. Thirdly, we propose an execution manner of iterative correction for language model which can effectively alleviate the impact of noise input. Additionally, based on the ensemble of iterative predictions, we propose a self-training method which can learn from unlabeled images effectively. Extensive experiments indicate that ABINet has superiority on low-quality images and achieves state-of-the-art results on several mainstream benchmarks. Besides, the ABINet trained with ensemble self-training shows promising improvement in realizing human-level recognition.

30.1.2 Dataset

Train Dataset

Test Dataset

30.1.3 Results and models

Coming Soon!

注解:

1. ABINet allows its encoder to run and be trained without decoder and fuser. Its encoder is designed to recognize texts as a stand-alone model and therefore can work as an independent text recognizer. We release it as ABINet-Vision.
2. Facts about the pretrained model: MMOCR does not have a systematic pipeline to pretrain the language model (LM) yet, thus the weights of LM are converted from the official pretrained model. The weights of ABINet-Vision are directly used as the vision model of ABINet.

30.1.4 Citation

```
@article{fang2021read,
  title={Read Like Humans: Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition},
  author={Fang, Shancheng and Xie, Hongtao and Wang, Yuxin and Mao, Zhendong and Zhang, Yongdong},
    booktitle={Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition},
  year={2021}
}
```

30.2 CRNN

An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition

30.2.1 Abstract

Image-based sequence recognition has been a long-standing research topic in computer vision. In this paper, we investigate the problem of scene text recognition, which is among the most important and challenging tasks in image-based sequence recognition. A novel neural network architecture, which integrates feature extraction, sequence modeling and transcription into a unified framework, is proposed. Compared with previous systems for scene text recognition, the proposed architecture possesses four distinctive properties: (1) It is end-to-end trainable, in contrast to most of the existing algorithms whose components are separately trained and tuned. (2) It naturally handles sequences in arbitrary lengths, involving no character segmentation or horizontal scale normalization. (3) It is not confined to any predefined lexicon and achieves remarkable performances in both lexicon-free and lexicon-based scene text recognition tasks. (4) It generates an effective yet much smaller model, which is more practical for real-world application scenarios. The experiments on standard benchmarks, including the IIIT-5K, Street View Text and ICDAR datasets, demonstrate the superiority of the proposed algorithm over the prior arts. Moreover, the proposed algorithm performs well in the task of image-based music score recognition, which evidently verifies the generality of it.

30.2.2 Dataset

Train Dataset

Test Dataset

30.2.3 Results and models

30.2.4 Citation

```
@article{shi2016end,
  title={An end-to-end trainable neural network for image-based sequence recognition
↪and its application to scene text recognition},
  author={Shi, Baoguang and Bai, Xiang and Yao, Cong},
  journal={IEEE transactions on pattern analysis and machine intelligence},
  year={2016}
}
```

30.3 MASTER

MASTER: Multi-aspect non-local network for scene text recognition

30.3.1 Abstract

Attention-based scene text recognizers have gained huge success, which leverages a more compact intermediate representation to learn 1d- or 2d- attention by a RNN-based encoder-decoder architecture. However, such methods suffer from attention-drift problem because high similarity among encoded features leads to attention confusion under the RNN-based local attention mechanism. Moreover, RNN-based methods have low efficiency due to poor parallelization. To overcome these problems, we propose the MASTER, a self-attention based scene text recognizer that (1) not only encodes the input-output attention but also learns self-attention which encodes feature-feature and target-target relationships inside the encoder and decoder and (2) learns a more powerful and robust intermediate representation to spatial distortion, and (3) owns a great training efficiency because of high training parallelization and a high-speed inference because of an efficient memory-cache mechanism. Extensive experiments on various benchmarks demonstrate the superior performance of our MASTER on both regular and irregular scene text.

30.3.2 Dataset

Train Dataset

Test Dataset

30.3.3 Results and Models

Coming Soon!

30.3.4 Citation

```
@article{Lu2021MASTER,
  title={{MASTER}: Multi-Aspect Non-local Network for Scene Text Recognition},
  author={Ning Lu and Wenwen Yu and Xianbiao Qi and Yihao Chen and Ping Gong and Rong␣
↪Xiao and Xiang Bai},
  journal={Pattern Recognition},
  year={2021}
}
```

30.4 NRTR

NRTR: A No-Recurrence Sequence-to-Sequence Model For Scene Text Recognition

30.4.1 Abstract

Scene text recognition has attracted a great many researches due to its importance to various applications. Existing methods mainly adopt recurrence or convolution based networks. Though have obtained good performance, these methods still suffer from two limitations: slow training speed due to the internal recurrence of RNNs, and high complexity due to stacked convolutional layers for long-term feature extraction. This paper, for the first time, proposes a no-recurrence sequence-to-sequence text recognizer, named NRTR, that dispenses with recurrences and convolutions entirely. NRTR follows the encoder-decoder paradigm, where the encoder uses stacked self-attention to extract image features, and the decoder applies stacked self-attention to recognize texts based on encoder output. NRTR relies solely on self-attention mechanism thus could be trained with more parallelization and less complexity. Considering scene image has large variation in text and background, we further design a modality-transform block to effectively transform 2D input images to 1D sequences, combined with the encoder to extract more discriminative features. NRTR achieves state-of-the-art or highly competitive performance on both regular and irregular benchmarks, while requires only a small fraction of training time compared to the best model from the literature (at least 8 times faster).

30.4.2 Dataset

Train Dataset

Test Dataset

30.4.3 Results and Models

Coming Soon!

30.4.4 Citation

```
@inproceedings{sheng2019nrtr,
  title={NRTR: A no-recurrence sequence-to-sequence model for scene text recognition},
  author={Sheng, Fenfen and Chen, Zhineng and Xu, Bo},
  booktitle={2019 International Conference on Document Analysis and Recognition␣
↪(ICDAR)},
  pages={781--786},
  year={2019},
  organization={IEEE}
}
```

30.5 RobustScanner

RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

30.5.1 Abstract

The attention-based encoder-decoder framework has recently achieved impressive results for scene text recognition, and many variants have emerged with improvements in recognition quality. However, it performs poorly on contextless texts (e.g., random character sequences) which is unacceptable in most of real application scenarios. In this paper, we first deeply investigate the decoding process of the decoder. We empirically find that a representative character-level sequence decoder utilizes not only context information but also positional information. Contextual information, which the existing approaches heavily rely on, causes the problem of attention drift. To suppress such side-effect, we propose a novel position enhancement branch, and dynamically fuse its outputs with those of the decoder attention module for scene text recognition. Specifically, it contains a position aware module to enable the encoder to output feature vectors encoding their own spatial positions, and an attention module to estimate glimpses using the positional clue (i.e., the current decoding time step) only. The dynamic fusion is conducted for more robust feature via an element-wise gate mechanism. Theoretically, our proposed method, dubbed \emph{RobustScanner}, decodes individual characters with dynamic ratio between context and positional clues, and utilizes more positional ones when the decoding sequences with scarce context, and thus is robust and practical. Empirically, it has achieved new state-of-the-art results on popular regular and irregular text recognition benchmarks while without much performance drop on contextless benchmarks, validating its robustness in both contextual and contextless application scenarios.

30.5.2 Dataset

Train Dataset

Test Dataset

30.5.3 Results and Models

Coming Soon!

30.5.4 References

[1] Li, Hui and Wang, Peng and Shen, Chunhua and Zhang, Guyu. Show, attend and read: A simple and strong baseline for irregular text recognition. In AAAI 2019.

30.5.5 Citation

```
@inproceedings{yue2020robustscanner,
  title={RobustScanner: Dynamically Enhancing Positional Clues for Robust Text
↪Recognition},
  author={Yue, Xiaoyu and Kuang, Zhanghui and Lin, Chenhao and Sun, Hongbin and Zhang,
↪ Wayne},
  booktitle={European Conference on Computer Vision},
  year={2020}
}
```

30.6 SAR

Show, Attend and Read: A Simple and Strong Baseline for Irregular Text Recognition

30.6.1 Abstract

Recognizing irregular text in natural scene images is challenging due to the large variance in text appearance, such as curvature, orientation and distortion. Most existing approaches rely heavily on sophisticated model designs and/or extra fine-grained annotations, which, to some extent, increase the difficulty in algorithm implementation and data collection. In this work, we propose an easy-to-implement strong baseline for irregular scene text recognition, using off-the-shelf neural network components and only word-level annotations. It is composed of a 31-layer ResNet, an LSTM-based encoder-decoder framework and a 2-dimensional attention module. Despite its simplicity, the proposed method is robust and achieves state-of-the-art performance on both regular and irregular scene text recognition benchmarks.

30.6.2 Dataset

Train Dataset

Test Dataset

30.6.3 Results and Models

Coming Soon!

30.6.4 Citation

```
@inproceedings{li2019show,
  title={Show, attend and read: A simple and strong baseline for irregular text recognition},
  author={Li, Hui and Wang, Peng and Shen, Chunhua and Zhang, Guyu},
  booktitle={Proceedings of the AAAI Conference on Artificial Intelligence},
  volume={33},
  number={01},
  pages={8610--8617},
  year={2019}
}
```

30.7 SATRN

On Recognizing Texts of Arbitrary Shapes with 2D Self-Attention

30.7.1 Abstract

Scene text recognition (STR) is the task of recognizing character sequences in natural scenes. While there have been great advances in STR methods, current methods still fail to recognize texts in arbitrary shapes, such as heavily curved or rotated texts, which are abundant in daily life (e.g. restaurant signs, product labels, company logos, etc). This paper introduces a novel architecture to recognizing texts of arbitrary shapes, named Self-Attention Text Recognition Network (SATRN), which is inspired by the Transformer. SATRN utilizes the self-attention mechanism to describe two-dimensional (2D) spatial dependencies of characters in a scene text image. Exploiting the full-graph propagation of self-attention, SATRN can recognize texts with arbitrary arrangements and large inter-character spacing. As a result, SATRN outperforms existing STR models by a large margin of 5.7 pp on average in "irregular text" benchmarks. We provide empirical analyses that illustrate the inner mechanisms and the extent to which the model is applicable (e.g. rotated and multi-line text). We will open-source the code.

30.7.2 Dataset

Train Dataset

Test Dataset

30.7.3 Results and Models

Coming Soon!

30.7.4 Citation

```
@article{junyeop2019recognizing,
  title={On Recognizing Texts of Arbitrary Shapes with 2D Self-Attention},
  author={Junyeop Lee, Sungrae Park, Jeonghun Baek, Seong Joon Oh, Seonghyeon Kim,
↪Hwalsuk Lee},
  year={2019}
}
```

CHAPTER 31

关键信息提取模型

31.1 SDMGR

Spatial Dual-Modality Graph Reasoning for Key Information Extraction

31.1.1 Abstract

Key information extraction from document images is of paramount importance in office automation. Conventional template matching based approaches fail to generalize well to document images of unseen templates, and are not robust against text recognition errors. In this paper, we propose an end-to-end Spatial Dual-Modality Graph Reasoning method (SDMG-R) to extract key information from unstructured document images. We model document images as dual-modality graphs, nodes of which encode both the visual and textual features of detected text regions, and edges of which represent the spatial relations between neighboring text regions. The key information extraction is solved by iteratively propagating messages along graph edges and reasoning the categories of graph nodes. In order to roundly evaluate our proposed method as well as boost the future research, we release a new dataset named WildReceipt, which is collected and annotated tailored for the evaluation of key information extraction from document images of unseen templates in the wild. It contains 25 key information categories, a total of about 69000 text boxes, and is about 2 times larger than the existing public datasets. Extensive experiments validate that all information including visual features, textual features and spatial relations can benefit key information extraction. It has been shown that SDMG-R can effectively extract key information from document images of unseen templates, and obtain new state-of-the-art results on the recent popular benchmark SROIE and our WildReceipt. Our code and dataset will be publicly released.

31.1.2 Results and models

WildReceipt

WildReceiptOpenset

31.1.3 Citation

```
@misc{sun2021spatial,
      title={Spatial Dual-Modality Graph Reasoning for Key Information Extraction},
      author={Hongbin Sun and Zhanghui Kuang and Xiaoyu Yue and Chenhao Lin and Wayne␣
↪Zhang},
      year={2021},
      eprint={2103.14470},
      archivePrefix={arXiv},
      primaryClass={cs.CV}
}
```

CHAPTER 32

贡献指南

CHAPTER 33

项目

CHAPTER 34

Changelog of v1.x

34.1 v1.0.0rc0 (1/9/2022)

We are excited to announce the release of MMOCR 1.0.0rc0. MMOCR 1.0.0rc0 is the first version of MMOCR 1.x, a part of the OpenMMLab 2.0 projects. Built upon the new training engine, MMOCR 1.x unifies the interfaces of dataset, models, evaluation, and visualization with faster training and testing speed.

34.1.1 Highlights

1. **New engines**. MMOCR 1.x is based on MMEngine, which provides a general and powerful runner that allows more flexible customizations and significantly simplifies the entrypoints of high-level interfaces.
2. **Unified interfaces**. As a part of the OpenMMLab 2.0 projects, MMOCR 1.x unifies and refactors the interfaces and internal logics of train, testing, datasets, models, evaluation, and visualization. All the OpenMMLab 2.0 projects share the same design in those interfaces and logics to allow the emergence of multi-task/modality algorithms.
3. **Cross project calling**. Benefiting from the unified design, you can use the models implemented in other OpenMMLab projects, such as MMDet. We provide an example of how to use MMDetection's Mask R-CNN through `MMDetWrapper`. Check our documents for more details. More wrappers will be released in the future.
4. **Stronger visualization**. We provide a series of useful tools which are mostly based on brand-new visualizers. As a result, it is more convenient for the users to explore the models and datasets now.
5. **More documentation and tutorials**. We add a bunch of documentation and tutorials to help users get started more smoothly. Read it here.

34.1.2 Breaking Changes

We briefly list the major breaking changes here. We will update the migration guide to provide complete details and migration instructions.

Dependencies

- MMOCR 1.x relies on MMEngine to run. MMEngine is a new foundational library for training deep learning models in OpenMMLab 2.0 models. The dependencies of file IO and training are migrated from MMCV 1.x to MMEngine.
- MMOCR 1.x relies on MMCV>=2.0.0rc0. Although MMCV no longer maintains the training functionalities since 2.0.0rc0, MMOCR 1.x relies on the data transforms, CUDA operators, and image processing interfaces in MMCV. Note that the package `mmcv` is the version that provide pre-built CUDA operators and `mmcv-lite` does not since MMCV 2.0.0rc0, while `mmcv-full` has been deprecated.

Training and testing

- MMOCR 1.x uses Runner in MMEngine rather than that in MMCV. The new Runner implements and unifies the building logic of dataset, model, evaluation, and visualizer. Therefore, MMOCR 1.x no longer maintains the building logics of those modules in `mmocr.train.apis` and `tools/train.py`. Those code have been migrated into MMEngine. Please refer to the migration guide of Runner in MMEngine for more details.
- The Runner in MMEngine also supports testing and validation. The testing scripts are also simplified, which has similar logic as that in training scripts to build the runner.
- The execution points of hooks in the new Runner have been enriched to allow more flexible customization. Please refer to the migration guide of Hook in MMEngine for more details.
- Learning rate and momentum scheduling has been migrated from `Hook` to `Parameter Scheduler` in MMEngine. Please refer to the migration guide of Parameter Scheduler in MMEngine for more details.

Configs

- The Runner in MMEngine uses a different config structures to ease the understanding of the components in runner. Users can read the *config example of MMOCR* or refer to the migration guide in MMEngine for migration details.
- The file names of configs and models are also refactored to follow the new rules unified across OpenMMLab 2.0 projects. Please refer to the *user guides of config* for more details.

Dataset

The Dataset classes implemented in MMOCR 1.x all inherits from the `BaseDetDataset`, which inherits from the BaseDataset in MMEngine. There are several changes of Dataset in MMOCR 1.x.

- All the datasets support to serialize the data list to reduce the memory when multiple workers are built to accelerate data loading.
- The interfaces are changed accordingly.

Data Transforms

The data transforms in MMOCR 1.x all inherits from those in MMCV>=2.0.0rc0, which follows a new convention in OpenMMLab 2.0 projects. The changes are listed as below:

- The interfaces are also changed. Please refer to the API Reference
- The functionality of some data transforms (e.g., `Resize`) are decomposed into several transforms.
- The same data transforms in different OpenMMLab 2.0 libraries have the same augmentation implementation and the logic of the same arguments, i.e., `Resize` in MMDet 3.x and MMOCR 1.x will resize the image in the exact same manner given the same arguments.

Model

The models in MMOCR 1.x all inherits from `BaseModel` in MMEngine, which defines a new convention of models in OpenMMLab 2.0 projects. Users can refer to the tutorial of model in MMengine for more details. Accordingly, there are several changes as the following:

- The model interfaces, including the input and output formats, are significantly simplified and unified following the new convention in MMOCR 1.x. Specifically, all the input data in training and testing are packed into `inputs` and `data_samples`, where `inputs` contains model inputs like a list of image tensors, and `data_samples` contains other information of the current data sample such as ground truths and model predictions. In this way, different tasks in MMOCR 1.x can share the same input arguments, which makes the models more general and suitable for multi-task learning.
- The model has a data preprocessor module, which is used to pre-process the input data of model. In MMOCR 1.x, the data preprocessor usually does necessary steps to form the input images into a batch, such as padding. It can also serve as a place for some special data augmentations or more efficient data transformations like normalization.
- The internal logic of model have been changed. In MMOCR 0.x, model used `forward_train` and `simple_test` to deal with different model forward logics. In MMOCR 1.x and OpenMMLab 2.0, the forward function has three modes: `loss`, `predict`, and `tensor` for training, inference, and tracing or other purposes, respectively. The forward function calls `self.loss()`, `self.predict()`, and `self._forward()` given the modes `loss`, `predict`, and `tensor`, respectively.

Evaluation

MMOCR 1.x mainly implements corresponding metrics for each task, which are manipulated by Evaluator to complete the evaluation. In addition, users can build evaluator in MMOCR 1.x to conduct offline evaluation, i.e., evaluate predictions that may not produced by MMOCR, prediction follows our dataset conventions. More details can be find in the Evaluation Tutorial in MMEngine.

Visualization

The functions of visualization in MMOCR 1.x are removed. Instead, in OpenMMLab 2.0 projects, we use Visualizer to visualize data. MMOCR 1.x implements `TextDetLocalVisualizer`, `TextRecogLocalVisualizer`, and `KIELocalVisualizer` to allow visualization of ground truths, model predictions, and feature maps, etc., at any place, for the three tasks supported in MMOCR. It also supports to dump the visualization data to any external visualization backends such as Tensorboard and Wandb. Check our Visualization Document for more details.

34.1.3 Improvements

- Most models enjoy a performance improvement from the new framework and refactor of data transforms. For example, in MMOCR 1.x, DBNet-R50 achieves **0.854** hmean score on ICDAR 2015, while the counterpart can only get **0.840** hmean score in MMOCR 0.x.
- Support mixed precision training of most of the models. However, the rest models are not supported yet because the operators they used might not be representable in fp16. We will update the documentation and list the results of mixed precision training.

34.1.4 Ongoing changes

1. Test-time augmentation: which was supported in MMOCR 0.x, is not implemented yet in this version due to limited time slot. We will support it in the following releases with a new and simplified design.
2. Inference interfaces: a unified inference interfaces will be supported in the future to ease the use of released models.
3. Interfaces of useful tools that can be used in notebook: more useful tools that implemented in the `tools/` directory will have their python interfaces so that they can be used through notebook and in downstream libraries.
4. Documentation: we will add more design docs, tutorials, and migration guidance so that the community can deep dive into our new design, participate the future development, and smoothly migrate downstream libraries to MMOCR 1.x.

CHAPTER 35

常见问题

CHAPTER 36

mmocr.datasets

class mmocr.datasets.**ConcatDataset**(*datasets*, *pipeline=[]*, *verify_meta=True*, *force_apply=False*, *lazy_init=False*)

A wrapper of concatenated dataset.

Same as `torch.utils.data.dataset.ConcatDataset` and support lazy_init.

注解: `ConcatDataset` should not inherit from `BaseDataset` since `get_subset` and `get_subset_` could produce ambiguous meaning sub-dataset which conflicts with original dataset. If you want to use a sub-dataset of `ConcatDataset`, you should set `indices` arguments for wrapped dataset which inherit from `BaseDataset`.

参数

- **datasets** (*Sequence[BaseDataset] or Sequence[dict]*) –A list of datasets which will be concatenated.
- **pipeline** (*list, optional*) –Processing pipeline to be applied to all of the concatenated datasets. Defaults to [].
- **verify_meta** (*bool*) –Whether to verify the consistency of meta information of the concatenated datasets. Defaults to True.
- **force_apply** (*bool*) –Whether to force apply pipeline to all datasets if any of them already has the pipeline configured. Defaults to False.

- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. Defaults to False.

class mmocr.datasets.**IcdarDataset**(**args*, *proposal_file=None*, *file_client_args={'backend': 'disk'}*, ***kwargs*)

Dataset for text detection while ann_file in coco format.

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str*) –The root directory for `data_prefix` and `ann_file`. Defaults to ''.
- **data_prefix** (*dict*) –Prefix for training data. Defaults to dict(img_path='').
- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `Basedataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.
- **max_refetch** (*int, optional*) –If `Basedataset.prepare_data` get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

parse_data_info(*raw_data_info*)

Parse raw annotation to target format.

参数 **raw_data_info** (*dict*) –Raw data information loaded from `ann_file`

返回 Parsed annotation.

返回类型 Union[dict, List[dict]]

class mmocr.datasets.**OCRDataset**(*ann_file=''*, *metainfo=None*, *data_root=''*, *data_prefix={'img_path': ''}*, *filter_cfg=None*, *indices=None*, *serialize_data=True*, *pipeline=[]*, *test_mode=False*, *lazy_init=False*, *max_refetch=1000*)

OCRDataset for text detection and text recognition.

The annotation format is shown as follows.

```
{
    "metainfo":
    {
      "dataset_type": "test_dataset",
      "task_name": "test_task"
    },
    "data_list":
    [
      {
        "img_path": "test_img.jpg",
        "height": 604,
        "width": 640,
        "instances":
        [
          {
            "bbox": [0, 0, 10, 20],
            "bbox_label": 1,
            "mask": [0,0,0,10,10,20,20,0],
            "text": '123'
          },
          {
            "bbox": [10, 10, 110, 120],
            "bbox_label": 2,
            "mask": [10,10],10,110,110,120,120,10]],
            "extra_anns": '456'
          }
        ]
      },
    ]
}
```

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str, optional*) –The root directory for `data_prefix` and `ann_file`. Defaults to None.

- **data_prefix** (*dict, optional*) –Prefix for training data. Defaults to dict(img=None, ann=None).
- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `OCRdataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.
- **max_refetch** (*int, optional*) –If `OCRdataset.prepare_data` get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

注解: OCRDataset collects meta information from *annotation file* (the lowest priority), "OCRDataset.METAINFO"(medium) and *metainfo parameter* (highest) passed to constructors. The lower priority meta information will be overwritten by higher one.

实际案例

Assume the annotation file is given above. »> class CustomDataset(OCRDataset): »> METAINFO: dict = dict(task_name='custom_task', »> dataset_type='custom_type') »> metainfo=dict(task_name='custom_task_name') »> custom_dataset = CustomDataset(»> 'path/to/ann_file', »> metainfo=metainfo) »> # meta information of annotation file will be overwritten by »> # *CustomDataset.METAINFO*. The merged meta information will »> # further be overwritten by argument *metainfo*. »> custom_dataset.metainfo { 'task_name': custom_task_name, dataset_type: custom_type}

class mmocr.datasets.**RecogLMDBDataset**(*ann_file=''*, *parser_cfg={'keys': ['filename', 'text'], 'type': 'LineJsonParser'}*, *metainfo=None*, *data_root=''*, *data_prefix={'img_path': ''}*, *filter_cfg=None*, *indices=None*, *serialize_data=True*, *pipeline=[]*, *test_mode=False*, *lazy_init=False*, *max_refetch=1000*)

RecogLMDBDataset for text recognition.

The annotation format should be in lmdb format. We support two lmdb formats, one is the lmdb file with only labels generated by txt2lmdb (deprecated), and another one is the lmdb file generated by recog2lmdb.

The former format stores string in *filename text* format directly in lmdb, while the latter uses *image_key* as well as *label_key* for querying.

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **parse_cfg** (*dict, optional*) –Config of parser for parsing annotations. Use `LineJsonParser` when the annotation file is in jsonl format with keys of `filename` and `text`. The keys in parse_cfg should be consistent with the keys in jsonl annotations. The first key in parse_cfg should be the key of the path in jsonl annotations. The second key in parse_cfg should be the key of the text in jsonl Use `LineStrParser` when the annotation file is in txt format. Defaults to `dict(type='LineJsonParser', keys=['filename', 'text'])`.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str*) –The root directory for `data_prefix` and `ann_file`. Defaults to ''.
- **data_prefix** (*dict*) –Prefix for training data. Defaults to `dict(img_path='')`.
- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `RecogLMDBDataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.
- **max_refetch** (*int, optional*) –If `RecogLMDBdataset.prepare_data` get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

close()

Close lmdb environment.

load_data_list()

Load annotations from an annotation file named as `self.ann_file`

返回 A list of annotation.

返回类型 List[dict]

parse_data_info(*raw_anno_info*)

Parse raw annotation to target format.

参数 **raw_anno_info** (*str*) –One raw data information loaded from `ann_file`.

返回 Parsed annotation.

返回类型 (dict)

class mmocr.datasets.**RecogTextDataset**(*ann_file=''*, *file_client_args=None*, *parser_cfg={'keys': ['filename', 'text'], 'type': 'LineJsonParser'}*, *metainfo=None*, *data_root=''*, *data_prefix={'img_path': ''}*, *filter_cfg=None*, *indices=None*, *serialize_data=True*, *pipeline=[]*, *test_mode=False*, *lazy_init=False*, *max_refetch=1000*)

RecogTextDataset for text recognition.

The annotation format can be both in jsonl and txt. If the annotation file is in jsonl format, it should be a list of dicts. If the annotation file is in txt format, it should be a list of lines.

The annotation formats are shown as follows. - txt format .. code-block:: none

```
test_img1.jpg OpenMMLab test_img2.jpg MMOCR
```

- jsonl format

```
``{"filename": "test_img1.jpg", "text": "OpenMMLab"}``
``{"filename": "test_img2.jpg", "text": "MMOCR"}``
```

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **file_client_args** (*dict, optional*) –Arguments to instantiate a FileClient. See `mmengine.fileio.FileClient` for details. Default: None.
- **parse_cfg** (*dict, optional*) –Config of parser for parsing annotations. Use `LineJsonParser` when the annotation file is in jsonl format with keys of `filename` and `text`. The keys in parse_cfg should be consistent with the keys in jsonl annotations. The first key in parse_cfg should be the key of the path in jsonl annotations. The second key in parse_cfg should be the key of the text in jsonl Use `LineStrParser` when the annotation file is in txt format. Defaults to `dict(type='LineJsonParser', keys=['filename', 'text'])`.

- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str*) –The root directory for `data_prefix` and `ann_file`. Defaults to ''.
- **data_prefix** (*dict*) –Prefix for training data. Defaults to `dict(img_path='')`.
- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `RecogTextDataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.
- **max_refetch** (*int, optional*) –If `RecogTextDataset.prepare_data` get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

load_data_list()

Load annotations from an annotation file named as `self.ann_file`

返回 A list of annotation.

返回类型 List[dict]

parse_data_info(*raw_anno_info*)

Parse raw annotation to target format.

参数 **raw_anno_info** (*str*) –One raw data information loaded from `ann_file`.

返回 Parsed annotation.

返回类型 (dict)

class mmocr.datasets.**WildReceiptDataset**(*directed=False*, *ann_file=''*, *metainfo=None*, *data_root=''*, *data_prefix={'img_path': ''}*, *filter_cfg=None*, *indices=None*, *serialize_data=True*, *pipeline=Ellipsis*, *test_mode=False*, *lazy_init=False*, *max_refetch=1000*)

WildReceipt Dataset for key information extraction. There are two files to be loaded: metainfo and annotation.

The metainfo file contains the mapping between classes and labels. The annotation file contains the all necessary information about the image, such as bounding boxes, texts, and labels etc.

The metainfo file is a text file with the following format:

```
0 Ignore
1 Store_name_value
2 Store_name_key
```

The annotation format is shown as follows.

```
{
    "file_name": "a.jpeg",
    "height": 348,
    "width": 348,
    "annotations": [
        {
            "box": [
                114.0,
                19.0,
                230.0,
                19.0,
                230.0,
                1.0,
                114.0,
                1.0
            ],
            "text": "CHOEUN",
            "label": 1
        },
        {
            "box": [
                97.0,
                35.0,
                236.0,
                35.0,
                236.0,
                19.0,
                97.0,
                19.0
            ],
            "text": "KOREANRESTAURANT",
            "label": 2
        }
    ]
}
```

参数

- **directed** (*bool*) –Whether to use directed graph. Defaults to False.
- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **metainfo** (*str or dict, optional*) –Meta information for dataset, such as class information. If it's a string, it will be treated as a path to the class file from which the class information will be loaded. Defaults to None.
- **data_root** (*str, optional*) –The root directory for `data_prefix` and `ann_file`. Defaults to ''.
- **data_prefix** (*dict, optional*) –Prefix for training data. Defaults to dict(img_path='').
- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `Basedataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.
- **max_refetch** (*int, optional*) –If `Basedataset.prepare_data` get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

load_data_list()

Load data list from annotation file.

返回 A list of annotation dict.

返回类型 List[dict]

parse_data_info(*raw_data_info*)

Parse data info from raw data info.

参数 **raw_data_info** (*dict*) –Raw data info.

返回

Parsed data info.

- img_path (str): Path to the image.
- img_shape (tuple(int, int)): Image shape in (H, W).
- instances (list[dict]): A list of instances. - bbox (ndarray(dtype=np.float32)): Shape (4,). Bounding box. - text (str): Annotation text. - edge_label (int): Edge label. - bbox_label (int): Bounding box label.

返回类型 dict

36.1 Dataset Types

class mmocr.datasets.ocr_dataset.**OCRDataset**(*ann_file=''*, *metainfo=None*, *data_root=''*, *data_prefix={'img_path': ''}*, *filter_cfg=None*, *indices=None*, *serialize_data=True*, *pipeline=[]*, *test_mode=False*, *lazy_init=False*, *max_refetch=1000*)

OCRDataset for text detection and text recognition.

The annotation format is shown as follows.

```
{
    "metainfo":
    {
      "dataset_type": "test_dataset",
      "task_name": "test_task"
    },
    "data_list":
    [
      {
        "img_path": "test_img.jpg",
        "height": 604,
        "width": 640,
        "instances":
        [
          {
            "bbox": [0, 0, 10, 20],
            "bbox_label": 1,
            "mask": [0,0,0,10,10,20,20,0],
            "text": '123'
          },
          {
            "bbox": [10, 10, 110, 120],
```

(下页继续)

(续上页)

```
            "bbox_label": 2,
            "mask": [10,10],10,110,110,120,120,10]],
            "extra_anns": '456'
          }
        ]
      },
    ]
}
```

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ‘’.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str, optional*) –The root directory for `data_prefix` and `ann_file`. Defaults to None.
- **data_prefix** (*dict, optional*) –Prefix for training data. Defaults to dict(img=None, ann=None).
- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `OCRdataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.
- **max_refetch** (*int, optional*) –If `OCRdataset.prepare_data` get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

注解: OCRDataset collects meta information from *annotation file* (the lowest priority), “OCRDataset.METAINFO“(medium) and *metainfo parameter* (highest) passed to constructors. The lower priority meta

information will be overwritten by higher one.

实际案例

Assume the annotation file is given above. »> class CustomDataset(OCRDataset): »> METAINFO: dict = dict(task_name='custom_task', »> dataset_type='custom_type') »> metainfo=dict(task_name='custom_task_name') »> custom_dataset = CustomDataset(»> 'path/to/ann_file', »> metainfo=metainfo) »> # meta information of annotation file will be overwritten by »> # *CustomDataset.METAINFO*. The merged meta information will »> # further be overwritten by argument *metainfo*. »> custom_dataset.metainfo {'task_name': custom_task_name, dataset_type: custom_type}

class mmocr.datasets.icdar_dataset.**IcdarDataset**(**args*, *proposal_file=None*, *file_client_args={'backend': 'disk'}*, ***kwargs*)

Dataset for text detection while ann_file in coco format.

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str*) –The root directory for `data_prefix` and `ann_file`. Defaults to ''.
- **data_prefix** (*dict*) –Prefix for training data. Defaults to dict(img_path='').
- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `Basedataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.

- **max_refetch** (*int, optional*) –If Basedataset.prepare_data get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

parse_data_info(*raw_data_info*)

Parse raw annotation to target format.

参数 **raw_data_info** (*dict*) –Raw data information loaded from ann_file

返回 Parsed annotation.

返回类型 Union[dict, List[dict]]

class mmocr.datasets.recog_lmdb_dataset.**RecogLMDBDataset**(*ann_file=''*, *parser_cfg={'keys': ['filename', 'text'], 'type': 'LineJsonParser'}*, *metainfo=None*, *data_root=''*, *data_prefix={'img_path': ''}*, *filter_cfg=None*, *indices=None*, *serialize_data=True*, *pipeline=[]*, *test_mode=False*, *lazy_init=False*, *max_refetch=1000*)

RecogLMDBDataset for text recognition.

The annotation format should be in lmdb format. We support two lmdb formats, one is the lmdb file with only labels generated by txt2lmdb (deprecated), and another one is the lmdb file generated by recog2lmdb.

The former format stores string in *filename text* format directly in lmdb, while the latter uses *image_key* as well as *label_key* for querying.

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **parse_cfg** (*dict, optional*) –Config of parser for parsing annotations. Use LineJsonParser when the annotation file is in jsonl format with keys of filename and text. The keys in parse_cfg should be consistent with the keys in jsonl annotations. The first key in parse_cfg should be the key of the path in jsonl annotations. The second key in parse_cfg should be the key of the text in jsonl Use LineStrParser when the annotation file is in txt format. Defaults to dict(type='LineJsonParser', keys=['filename', 'text']).
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str*) –The root directory for data_prefix and ann_file. Defaults to ''.
- **data_prefix** (*dict*) –Prefix for training data. Defaults to dict(img_path='').

- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `RecogLMDBDataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.
- **max_refetch** (*int, optional*) –If `RecogLMDBdataset.prepare_data` get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

close()

Close lmdb environment.

load_data_list()

Load annotations from an annotation file named as `self.ann_file`

返回 A list of annotation.

返回类型 List[dict]

parse_data_info(*raw_anno_info*)

Parse raw annotation to target format.

参数 **raw_anno_info** (*str*) –One raw data information loaded from `ann_file`.

返回 Parsed annotation.

返回类型 (dict)

class mmocr.datasets.recog_text_dataset.**RecogTextDataset**(*ann_file=''*, *file_client_args=None*, *parser_cfg={'keys': ['filename', 'text'], 'type': 'LineJsonParser'}*, *metainfo=None*, *data_root=''*, *data_prefix={'img_path': ''}*, *filter_cfg=None*, *indices=None*, *serialize_data=True*, *pipeline=[]*, *test_mode=False*, *lazy_init=False*, *max_refetch=1000*)

RecogTextDataset for text recognition.

The annotation format can be both in jsonl and txt. If the annotation file is in jsonl format, it should be a list of dicts. If the annotation file is in txt format, it should be a list of lines.

The annotation formats are shown as follows. - txt format .. code-block:: none

```
test_img1.jpg OpenMMLab test_img2.jpg MMOCR
```

- jsonl format

```
``{"filename": "test_img1.jpg", "text": "OpenMMLab"}``
``{"filename": "test_img2.jpg", "text": "MMOCR"}``
```

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **file_client_args** (*dict, optional*) –Arguments to instantiate a FileClient. See mmengine.fileio.FileClient for details. Default: None.
- **parse_cfg** (*dict, optional*) –Config of parser for parsing annotations. Use LineJsonParser when the annotation file is in jsonl format with keys of filename and text. The keys in parse_cfg should be consistent with the keys in jsonl annotations. The first key in parse_cfg should be the key of the path in jsonl annotations. The second key in parse_cfg should be the key of the text in jsonl Use LineStrParser when the annotation file is in txt format. Defaults to dict(type='LineJsonParser', keys=['filename', 'text']).
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str*) –The root directory for data_prefix and ann_file. Defaults to ''.
- **data_prefix** (*dict*) –Prefix for training data. Defaults to dict(img_path='').

- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `RecogTextDataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.
- **max_refetch** (*int, optional*) –If `RecogTextDataset.prepare_data` get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

load_data_list()

Load annotations from an annotation file named as `self.ann_file`

返回 A list of annotation.

返回类型 List[dict]

parse_data_info(*raw_anno_info*)

Parse raw annotation to target format.

参数 **raw_anno_info** (*str*) –One raw data information loaded from `ann_file`.

返回 Parsed annotation.

返回类型 (dict)

class mmocr.datasets.wildreceipt_dataset.**WildReceiptDataset**(*directed=False*, *ann_file=''*, *metainfo=None*, *data_root=''*, *data_prefix={'img_path': ''}*, *filter_cfg=None*, *indices=None*, *serialize_data=True*, *pipeline=Ellipsis*, *test_mode=False*, *lazy_init=False*, *max_refetch=1000*)

WildReceipt Dataset for key information extraction. There are two files to be loaded: metainfo and annotation. The metainfo file contains the mapping between classes and labels. The annotation file contains the all necessary information about the image, such as bounding boxes, texts, and labels etc.

The metainfo file is a text file with the following format:

```
0 Ignore
1 Store_name_value
2 Store_name_key
```

The annotation format is shown as follows.

```
{
    "file_name": "a.jpeg",
    "height": 348,
    "width": 348,
    "annotations": [
        {
            "box": [
                114.0,
                19.0,
                230.0,
                19.0,
                230.0,
                1.0,
                114.0,
                1.0
            ],
            "text": "CHOEUN",
            "label": 1
        },
        {
            "box": [
```

(下页继续)

(续上页)

```
                97.0,
                35.0,
                236.0,
                35.0,
                236.0,
                19.0,
                97.0,
                19.0
            ],
            "text": "KOREANRESTAURANT",
            "label": 2
        }
    ]
}
```

参数

- **directed** (*bool*) –Whether to use directed graph. Defaults to False.
- **ann_file** (*str*) –Annotation file path. Defaults to ‘’.
- **metainfo** (*str or dict, optional*) –Meta information for dataset, such as class information. If it’s a string, it will be treated as a path to the class file from which the class information will be loaded. Defaults to None.
- **data_root** (*str, optional*) –The root directory for `data_prefix` and `ann_file`. Defaults to ‘’.
- **data_prefix** (*dict, optional*) –Prefix for training data. Defaults to dict(img_path=’’).
- **filter_cfg** (*dict, optional*) –Config for filter data. Defaults to None.
- **indices** (*int or Sequence[int], optional*) –Support using first few data in annotation file to facilitate training/testing on a smaller dataset. Defaults to None which means using all `data_infos`.
- **serialize_data** (*bool, optional*) –Whether to hold memory using serialized objects, when enabled, data loader workers can use shared RAM from master process instead of making a copy. Defaults to True.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **lazy_init** (*bool, optional*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is

not necessary to load annotation file. `Basedataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.

- **max_refetch** (*int, optional*) –If `Basedataset.prepare_data` get a None img. The maximum extra number of cycles to get a valid image. Defaults to 1000.

load_data_list()

Load data list from annotation file.

返回 A list of annotation dict.

返回类型 List[dict]

parse_data_info(*raw_data_info*)

Parse data info from raw data info.

参数 **raw_data_info** (*dict*) –Raw data info.

返回

Parsed data info.

- img_path (str): Path to the image.
- img_shape (tuple(int, int)): Image shape in (H, W).
- instances (list[dict]): A list of instances. - bbox (ndarray(dtype=np.float32)): Shape (4,). Bounding box. - text (str): Annotation text. - edge_label (int): Edge label. - bbox_label (int): Bounding box label.

返回类型 dict

36.2 Transforms

class mmocr.datasets.transforms.**BoundedScaleAspectJitter**(*long_size_bound*, *short_size_bound*, *ratio_range=(0.7, 1.3)*, *aspect_ratio_range=(0.9, 1.1)*, *resize_type='Resize'*, ***resize_kwargs*)

First randomly rescale the image so that the longside and shortside of the image are around the bound; then jitter its aspect ratio.

Required Keys:

- img
- img_shape
- gt_bboxes (optional)

- gt_polygons (optional)

Modified Keys:

- img
- img_shape
- gt_bboxes (optional)
- gt_polygons (optional)

Added Keys:

- scale
- scale_factor
- keep_ratio

参数

- **long_size_bound** (*int*) –The approximate bound for long size.
- **short_size_bound** (*int*) –The approximate bound for short size.
- **size_jitter_range** (*tuple(float, float)*) –Range of the ratio used to jitter the size. Defaults to (0.7, 1.3).
- **aspect_ratio_jitter_range** (*tuple(float, float)*) –Range of the ratio used to jitter its aspect ratio. Defaults to (0.9, 1.1).
- **resize_type** (*str*) –The type of resize class to use. Defaults to "Resize".
- ****resize_kwargs** –Other keyword arguments for the `resize_type`.
- **ratio_range** (*Tuple[float, float]*) –
- **aspect_ratio_range** (*Tuple[float, float]*) –

返回类型 None

transform(*results*)

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

class mmocr.datasets.transforms.**FixInvalidPolygon**(*mode='fix'*, *min_poly_points=3*)

Fix invalid polygons in the dataset.

Required Keys:

- gt_polygons
- gt_ignored

Modified Keys:

- gt_polygons
- gt_ignored

参数

- **mode** (*str*) –The mode of fixing invalid polygons. Options are ‘fix’ and ‘ignore’. For the ‘fix’ mode, the transform will try to fix the invalid polygons to a valid one by eliminating the self-intersection. For the ‘ignore’ mode, the invalid polygons will be ignored during training. Defaults to ‘fix’.
- **min_poly_points** (*int*) –Minimum number of the coordinate points in a polygon. Defaults to 3.

返回类型 None

transform(*results*)

Fix invalid polygons.

参数 **results** (*dict*) –Result dict containing the data to transform.

返回 The transformed data.

返回类型 dict

class mmocr.datasets.transforms.**ImgAugWrapper**(*args=None*)

A wrapper around imgaug https://github.com/aleju/imgaug.

Find available augmenters at https://imgaug.readthedocs.io/en/latest/source/overview_of_augmenters.html.

Required Keys:

- img
- gt_polygons (optional for text recognition)
- gt_bboxes (optional for text recognition)
- gt_bboxes_labels (optional for text recognition)
- gt_ignored (optional for text recognition)
- gt_texts (optional)

Modified Keys:

- img
- gt_polygons (optional for text recognition)
- gt_bboxes (optional for text recognition)
- gt_bboxes_labels (optional for text recognition)
- gt_ignored (optional for text recognition)
- img_shape (optional)
- gt_texts (optional)

参数 **args** (*list[list or dict], optional*) –The argumentation list. For details, please refer to imgaug document. Take args=[[‘Fliplr’, 0.5], dict(cls=’Affine’, rotate=[-10, 10]), [‘Resize’, [0.5, 3.0]]] as an example. The args horizontally flip images with probability 0.5, followed by random rotation with angles in range [-10, 10], and resize with an independent scale in range [0.5, 3.0] for each side of images. Defaults to None.

返回类型 None

transform(*results*)

Transform the image and annotation data.

参数 **results** (*dict*) –Result dict containing the data to transform.

返回 The transformed data.

返回类型 dict

class mmocr.datasets.transforms.**LoadImageFromFile**(*to_float32=False*, *color_type='color'*, *imdecode_backend='cv2'*, *file_client_args={'backend': 'disk'}*, *min_size=0*, *ignore_empty=False*)

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 :func:mmcv.imfrombytes. Defaults to 'color'.
- **imdecode_backend** (*str*) –The image decoding backend type. The backend argument for :func:mmcv.imfrombytes. See :func:mmcv.imfrombytes for details. Defaults to 'cv2'.
- **file_client_args** (*dict*) –Arguments to instantiate a FileClient. See mmengine.fileio.FileClient for details. Defaults to dict(backend='disk').
- **ignore_empty** (*bool*) –Whether to allow loading empty image or file path not existent. Defaults to False.
- **min_size** (*int*) –The minimum size of the image to be loaded. If the image is smaller than the minimum size, it will be regarded as a broken image. Defaults to 0.

返回类型 None

transform(*results*)

Functions to load image.

参数 **results** (*dict*) –Result dict from :obj:mmcv.BaseDataset.

返回类型 Optional[dict]

class mmocr.datasets.transforms.**LoadImageFromLMDB**(*to_float32=False*, *color_type='color'*, *imdecode_backend='cv2'*, *file_client_args={}*, *ignore_empty=False*)

Load an image from lmdb file. Only support LMDB file at disk.

LMDB file is organized with the following structure:

lmdb |__data.mdb |__lock.mdb

Required Keys:

- img_path (In LMDB img_path is a key in the format of "image-{i:09d}".)

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 :func:mmcv.imfrombytes. Defaults to 'color'.
- **imdecode_backend** (*str*) –The image decoding backend type. The backend argument for :func:mmcv.imfrombytes. See :func:mmcv.imfrombytes for details. Defaults to 'cv2'.
- **file_client_args** (*dict*) –Arguments to instantiate a FileClient except for backend and db_path. See mmengine.fileio.FileClient for details. Defaults to dict().
- **ignore_empty** (*bool*) –Whether to allow loading empty image or file path not existent. Defaults to False.

返回类型 None

transform(*results*)

Functions to load image from LMDB file.

参数 **results** (*dict*) –Result dict from :obj:mmcv.BaseDataset.

返回 The dict contains loaded image and meta information.

返回类型 dict

class mmocr.datasets.transforms.**LoadImageFromNDArray**(*to_float32=False*, *color_type='color'*, *imdecode_backend='cv2'*, *file_client_args={'backend': 'disk'}*, *min_size=0*, *ignore_empty=False*)

Load an image from results['img'].

Similar with *LoadImageFromFile*, but the image has been loaded as np.ndarray in results['img']. Can be used when loading image from webcam.

Required Keys:

- img

Modified Keys:

- img
- img_path
- 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*) –

- **imdecode_backend** (*str*) –
- **file_client_args** (*dict*) –
- **min_size** (*int*) –
- **ignore_empty** (*bool*) –

返回类型 None

transform(*results*)

Transform function to add image meta information.

参数 **results** (*dict*) –Result dict with Webcam read image in `results['img']`.

返回 The dict contains loaded image and meta information.

返回类型 dict

class mmocr.datasets.transforms.**LoadKIEAnnotations**(*with_bbox=True*, *with_label=True*, *with_text=True*, *directed=False*, *key_node_idx=None*, *value_node_idx=None*, ***kwargs*)

Load and process the `instances` annotation provided by dataset.

The annotation format is as the following:

```
{
    # A nested list of 4 numbers representing the bounding box of the
    # instance, in (x1, y1, x2, y2) order.
    'bbox': np.array([[x1, y1, x2, y2], [x1, y1, x2, y2], ...],
                     dtype=np.int32),

    # Labels of boxes. Shape is (N,).
    'bbox_labels': np.array([0, 2, ...], dtype=np.int32),

    # Labels of edges. Shape (N, N).
    'edge_labels': np.array([0, 2, ...], dtype=np.int32),

    # List of texts.
    "texts": ['text1', 'text2', ...],
}
```

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),
```

(下页继续)

(续上页)

```
    # In np.int64 type.
    'gt_bboxes_labels': np.ndarray(N, ),
    # In np.int32 type.
    'gt_edges_labels': np.ndarray(N, N),
    # In list[str]
    'gt_texts': list[str],
    # tuple(int)
    'ori_shape': (H, W)
}
```

Required Keys:

- bboxes
- bbox_labels
- edge_labels
- texts

Added Keys:

- gt_bboxes (np.float32)
- gt_bboxes_labels (np.int64)
- gt_edges_labels (np.int64)
- gt_texts (list[str])
- ori_shape (tuple[int])

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_text** (*bool*) –Whether to parse and load the text annotation. Defaults to True.
- **directed** (*bool*) –Whether build edges as a directed graph. Defaults to False.
- **key_node_idx** (*int, optional*) –Key node label, used to mask out edges that are not connected from key nodes to value nodes. It has to be specified together with `value_node_idx`. Defaults to None.
- **value_node_idx** (*int, optional*) –Value node label, used to mask out edges that are not connected from key nodes to value nodes. It has to be specified together with `key_node_idx`. Defaults to None.

返回类型 None

transform(*results*)

Function to load multiple types annotations.

参数 **results** (*dict*) –Result dict from :obj:OCRDataset.

返回 The dict contains loaded bounding box, label polygon and text annotations.

返回类型 dict

class mmocr.datasets.transforms.**LoadOCRAnnotations**(*with_bbox=False*, *with_label=False*, *with_polygon=False*, *with_text=False*, ***kwargs*)

Load and process the `instances` annotation provided by dataset.

The annotation format is as the following:

```
{
    'instances':
    [
        {
        # List of 4 numbers representing the bounding box of the
        # instance, in (x1, y1, x2, y2) order.
        # used in text detection or text spotting tasks.
        'bbox': [x1, y1, x2, y2],

        # Label of instance, usually it's 0.
        # used in text detection or text spotting tasks.
        'bbox_label': 0,

        # List of n numbers representing the polygon of the
        # instance, in (xn, yn) order.
        # used in text detection/ textspotter.
        "polygon": [x1, y1, x2, y2, ... xn, yn],

        # The flag indicating whether the instance should be ignored.
        # used in text detection or text spotting tasks.
        "ignore": False,

        # The groundtruth of text.
        # used in text recognition or text spotting tasks.
        "text": 'tmp',
        }
    ]
}
```

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 (x1, y1,..., xk, yk) order, float type.
    # in list[np.float32]
    'gt_polygons': list[np.ndarray(2k, )]
     # In np.bool_ type.
    'gt_ignored': np.ndarray(N, )
     # In list[str]
    'gt_texts': list[str]
}
```

Required Keys:

- instances
  - bbox (optional)
  - bbox_label (optional)
  - polygon (optional)
  - ignore (optional)
  - text (optional)

Added Keys:

- gt_bboxes (np.float32)
- gt_bboxes_labels (np.int64)
- gt_polygons (list[np.float32])
- gt_ignored (**np.bool_**)
- gt_texts (list[str])

参数

- **with_bbox** (*bool*) –Whether to parse and load the bbox annotation. Defaults to False.
- **with_label** (*bool*) –Whether to parse and load the label annotation. Defaults to False.
- **with_polygon** (*bool*) –Whether to parse and load the polygon annotation. Defaults to False.
- **with_text** (*bool*) –Whether to parse and load the text annotation. Defaults to False.

返回类型 None

transform(*results*)

Function to load multiple types annotations.

参数 **results** (*dict*) –Result dict from :obj:OCRDataset.

返回 The dict contains loaded bounding box, label polygon and text annotations.

返回类型 dict

class mmocr.datasets.transforms.**MMDet2MMOCR**

Convert transforms' s data format from MMDet to MMOCR.

Required Keys:

- gt_masks (PolygonMasks | BitmapMasks) (optional)
- gt_ignore_flags (np.bool) (optional)

Added Keys:

- gt_polygons (list[np.ndarray])
- gt_ignored (np.ndarray)

transform(*results*)

Convert MMDet' s data format to MMOCR' s data format.

参数 **results** (*Dict*) –Result dict containing the data to transform.

返回 The transformed data.

返回类型 (Dict)

class mmocr.datasets.transforms.**MMOCR2MMDet**(*poly2mask=False*)

Convert transforms' s data format from MMOCR to MMDet.

Required Keys:

- img_shape
- gt_polygons (List[ndarray]) (optional)
- gt_ignored (np.bool) (optional)

Added Keys:

- gt_masks (PolygonMasks | BitmapMasks) (optional)
- gt_ignore_flags (np.bool) (optional)

参数 **poly2mask** (*bool*) –Whether to convert mask to bitmap. Default: True.

返回类型 None

transform(*results*)

Convert MMOCR' s data format to MMDet' s data format.

参数 **results** (*Dict*) –Result dict containing the data to transform.

返回 The transformed data.

返回类型 (Dict)

class mmocr.datasets.transforms.**PackKIEInputs**(*meta_keys=()*)

Pack the inputs data for key information extraction.

The type of outputs is *dict*:

- inputs: image converted to tensor, whose shape is (C, H, W).
- data_samples: Two components of `TextDetDataSample` will be updated:
  - gt_instances (InstanceData): Depending on annotations, a subset of the following keys will be updated:
    * bboxes (torch.Tensor((N, 4), dtype=torch.float32)): The groundtruth of bounding boxes in the form of [x1, y1, x2, y2]. Renamed from ‘gt_bboxes’.
    * labels (torch.LongTensor(N)): The labels of instances. Renamed from ‘gt_bboxes_labels’.
    * edge_labels (torch.LongTensor(N, N)): The edge labels. Renamed from ‘gt_edges_labels’.
    * texts (list[str]): The groundtruth texts. Renamed from ‘gt_texts’.
  - metainfo (dict): ‘metainfo’ is always populated. The contents of the ‘metainfo’ depends on `meta_keys`. By default it includes:
    * “img_path”: Path to the image file.
    * “img_shape”: Shape of the image input to the network as a tuple (h, w). Note that the image may be zero-padded afterward on the bottom/right if the batch tensor is larger than this shape.
    * “scale_factor”: A tuple indicating the ratio of width and height of the preprocessed image to the original one.
    * “ori_shape”: Shape of the preprocessed image as a tuple (h, w).

参数 **meta_keys** (*Sequence[str], optional*) –Meta keys to be converted to the metainfo of `TextDetSample`. Defaults to `('img_path', 'ori_shape', 'img_shape', 'scale_factor', 'flip', 'flip_direction')`.

transform(*results*)

Method to pack the input data.

参数 **results** (*dict*) –Result dict from the data pipeline.

返回

- ‘inputs’ (obj:*torch.Tensor*): Data for model forwarding.
- ’ data_samples’ (obj:*DetDataSample*): The annotation info of the sample.

返回类型 dict

class mmocr.datasets.transforms.**PackTextDetInputs**(*meta_keys=('img_path', 'ori_shape', 'img_shape', 'scale_factor', 'flip', 'flip_direction')*)

Pack the inputs data for text detection.

The type of outputs is *dict*:

- inputs: image converted to tensor, whose shape is (C, H, W).
- data_samples: Two components of `TextDetDataSample` will be updated:
  - gt_instances (InstanceData): Depending on annotations, a subset of the following keys will be updated:
    - bboxes (torch.Tensor((N, 4), dtype=torch.float32)): The groundtruth of bounding boxes in the form of [x1, y1, x2, y2]. Renamed from 'gt_bboxes'.
    - labels (torch.LongTensor(N)): The labels of instances. Renamed from 'gt_bboxes_labels'.
    - polygons(list[np.array((2k,), dtype=np.float32)]): The groundtruth of polygons in the form of [x1, y1,⋯, xk, yk]. Each element in polygons may have different number of points. Renamed from 'gt_polygons'. Using numpy instead of tensor is that polygon usually is not the output of model and operated on cpu.
    - ignored (torch.BoolTensor((N,))): The flag indicating whether the corresponding instance should be ignored. Renamed from 'gt_ignored'.
    - texts (list[str]): The groundtruth texts. Renamed from 'gt_texts'.
  - metainfo (dict): 'metainfo' is always populated. The contents of the 'metainfo' depends on `meta_keys`. By default it includes:
    - "img_path": Path to the image file.
    - "img_shape": Shape of the image input to the network as a tuple (h, w). Note that the image may be zero-padded afterward on the bottom/right if the batch tensor is larger than this shape.
    - "scale_factor": A tuple indicating the ratio of width and height of the preprocessed image to the original one.
    - "ori_shape": Shape of the preprocessed image as a tuple (h, w).
    - "pad_shape": Image shape after padding (if any Pad-related transform involved) as a tuple (h, w).
    - "flip": A boolean indicating if the image has been flipped.
    - `flip_direction`: the flipping direction.

参数 **meta_keys** (*Sequence[str], optional*) –Meta keys to be converted to the metainfo of `TextDetSample`. Defaults to `('img_path', 'ori_shape', 'img_shape', 'scale_factor', 'flip', 'flip_direction')`.

transform(*results*)

Method to pack the input data.

参数 results (*dict*) –Result dict from the data pipeline.

返回

- ‘inputs’ (obj:*torch.Tensor*): Data for model forwarding.
- ’ data_samples’ (obj:*DetDataSample*): The annotation info of the sample.

返回类型 dict

class mmocr.datasets.transforms.**PackTextRecogInputs**(*meta_keys=('img_path', 'ori_shape', 'img_shape', 'pad_shape', 'valid_ratio')*)

Pack the inputs data for text recognition.

The type of outputs is *dict*:

- inputs: Image as a tensor, whose shape is (C, H, W).
- data_samples: Two components of `TextRecogDataSample` will be updated:
  - gt_text (LabelData):
    - item(str): The groundtruth of text. Rename from ‘gt_texts’.
  - metainfo (dict): ‘metainfo’ is always populated. The contents of the ‘metainfo’ depends on `meta_keys`. By default it includes:
    - “img_path”: Path to the image file.
    - “ori_shape”: Shape of the preprocessed image as a tuple (h, w).
    - “img_shape”: Shape of the image input to the network as a tuple (h, w). Note that the image may be zero-padded afterward on the bottom/right if the batch tensor is larger than this shape.
    - “valid_ratio”: The proportion of valid (unpadded) content of image on the x-axis. It defaults to 1 if not set in pipeline.

参数 meta_keys (*Sequence[str], optional*) –Meta keys to be converted to the metainfo of `TextRecogDataSampel`. Defaults to `('img_path', 'ori_shape', 'img_shape', 'pad_shape', 'valid_ratio')`.

transform(*results*)

Method to pack the input data.

参数 results (*dict*) –Result dict from the data pipeline.

返回

- ‘inputs’ (obj:*torch.Tensor*): Data for model forwarding.
- **’ data_samples’ (obj:*TextRecogDataSample*): The annotation info** of the sample.

返回类型 dict

class mmocr.datasets.transforms.**PadToWidth**(*width*, *pad_cfg={'type': 'Pad'}*)

Only pad the image's width.

Required Keys:

- img

Modified Keys:

- img
- img_shape

Added Keys:

- pad_shape
- pad_fixed_size
- pad_size_divisor
- valid_ratio

参数

- **width** (*int*) –Target width of padded image. Defaults to None.
- **pad_cfg** (*dict*) –Config to construct the Resize transform. Refer to `Pad` for detail. Defaults to `dict(type='Pad')`.

返回类型 None

transform(*results*)

Call function to pad images.

参数 **results** (*dict*) –Result dict from loading pipeline.

返回 Updated result dict.

返回类型 dict

class mmocr.datasets.transforms.**PyramidRescale**(*factor=4*, *base_shape=(128, 512)*, *randomize_factor=True*)

Resize the image to the base shape, downsample it with gaussian pyramid, and rescale it back to original size.

Adapted from https://github.com/FangShancheng/ABINet.

Required Keys:

- img (ndarray)

Modified Keys:

- img (ndarray)

参数

- **factor** (*int*) –The decay factor from base size, or the number of downsampling operations from the base layer.
- **base_shape** (*tuple[int, int]*) –The shape (width, height) of the base layer of the pyramid.
- **randomize_factor** (*bool*) –If True, the final factor would be a random integer in [0, factor].

返回类型 None

transform(*results*)

Applying pyramid rescale on results.

参数 **results** (*dict*) –Result dict containing the data to transform.

返回 The transformed data.

返回类型 Dict

class mmocr.datasets.transforms.**RandomCrop**(*min_side_ratio=0.4*)

Randomly crop images and make sure to contain at least one intact instance.

Required Keys:

- img
- gt_polygons
- gt_bboxes
- gt_bboxes_labels
- gt_ignored
- gt_texts (optional)

Modified Keys:

- img
- img_shape
- gt_polygons
- gt_bboxes
- gt_bboxes_labels
- gt_ignored
- gt_texts (optional)

参数 **min_side_ratio** (*float*) –The ratio of the shortest edge of the cropped image to the original image size.

返回类型 None

transform(*results*)

Applying random crop on results. :param results: Result dict contains the data to transform. :type results: dict

返回 The transformed data.

返回类型 dict

参数 **results** (*Dict*) –

class mmocr.datasets.transforms.**RandomFlip**(*prob=None*, *direction='horizontal'*)

Flip the image & bbox polygon.

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_polygons (optional)

Modified Keys:

- img
- gt_bboxes (optional)
- gt_polygons (optional)

Added Keys:

- flip

- flip_direction

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'.

返回类型 None

flip_polygons(*polygons*, *img_shape*, *direction*)

Flip polygons horizontally, vertically or diagonally.

参数

- **polygons** (*list[numpy.ndarray*) –polygons.
- **img_shape** (*tuple[int]*) –Image shape (height, width)
- **direction** (*str*) –Flip direction. Options are 'horizontal', 'vertical' and 'diagonal'.

返回 Flipped polygons.

返回类型 list[numpy.ndarray]

class mmocr.datasets.transforms.**RandomRotate**(*max_angle=10*, *pad_with_fixed_color=False*, *pad_value=(0, 0, 0)*, *use_canvas=False*)

Randomly rotate the image, boxes, and polygons. For recognition task, only the image will be rotated. If set `use_canvas` as True, the shape of rotated image might be modified based on the rotated angle size, otherwise, the image will keep the shape before rotation.

Required Keys:

- img
- img_shape
- gt_bboxes (optional)
- gt_polygons (optional)

Modified Keys:

- img
- img_shape (optional)
- gt_bboxes (optional)
- gt_polygons (optional)

Added Keys:

- rotated_angle

参数

- **max_angle** (*int*) –The maximum rotation angle (can be bigger than 180 or a negative). Defaults to 10.
- **pad_with_fixed_color** (*bool*) –The flag for whether to pad rotated image with fixed value. Defaults to False.
- **pad_value** (*tuple[int, int, int]*) –The color value for padding rotated image. Defaults to (0, 0, 0).
- **use_canvas** (*bool*) –Whether to create a canvas for rotated image. Defaults to False. If set true, the image shape may be modified.

返回类型 None

transform(*results*)

Applying random rotate on results.

参数

- **results** (*Dict*) –Result dict containing the data to transform.
- **center_shift** (*Tuple[int, int]*) –The shifting offset of the center point

返回 The transformed data

返回类型 dict

class mmocr.datasets.transforms.**RescaleToHeight**(*height*, *min_width=None*, *max_width=None*, *width_divisor=1*, *resize_type='Resize'*, ***resize_kwargs*)

Rescale the image to the height according to setting and keep the aspect ratio unchanged if possible. However, if any of `min_width`, `max_width` or `width_divisor` are specified, aspect ratio may still be changed to ensure the width meets these constraints.

Required Keys:

- img

Modified Keys:

- img
- img_shape

Added Keys:

- scale

- scale_factor
- keep_ratio

参数

- **height** (*int*) –Height of rescaled image.
- **min_width** (*int, optional*) –Minimum width of rescaled image. Defaults to None.
- **max_width** (*int, optional*) –Maximum width of rescaled image. Defaults to None.
- **width_divisor** (*int*) –The divisor of width size. Defaults to 1.
- **resize_type** (*str*) –The type of resize class to use. Defaults to "Resize".
- ****resize_kwargs** –Other keyword arguments for the `resize_type`.

返回类型 None

transform(*results*)

Transform function to resize images, bounding boxes and polygons.

参数 **results** (*dict*) –Result dict from loading pipeline.

返回 Resized results.

返回类型 dict

class mmocr.datasets.transforms.**Resize**(*scale=None*, *scale_factor=None*, *keep_ratio=False*, *clip_object_border=True*, *backend='cv2'*, *interpolation='bilinear'*)

Resize image & bboxes & polygons.

This transform resizes the input image according to `scale` or `scale_factor`. Bboxes and polygons 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
- img_shape
- gt_bboxes
- gt_polygons

Modified Keys:

- img
- img_shape
- gt_bboxes
- gt_polygons

Added Keys:

- scale
- scale_factor
- keep_ratio

参数

- **scale** (*int or tuple*) –Image scales for resizing. Defaults to None.
- **scale_factor** (*float or tuple[float, float]*) –Scale factors for resizing. It's either a factor applicable to both dimensions or in the form of (scale_w, scale_h). 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. 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'.

返回类型 None

transform(*results*)

Transform function to resize images, bounding boxes and polygons.

参数 **results** (*dict*) –Result dict from loading pipeline.

返回 Resized results, 'img', 'gt_bboxes', 'gt_polygons', 'scale', 'scale_factor', 'height', 'width', and 'keep_ratio' keys are updated in result dict.

返回类型 dict

class mmocr.datasets.transforms.**ShortScaleAspectJitter**(*short_size=736*, *ratio_range=(0.7, 1.3)*, *aspect_ratio_range=(0.9, 1.1)*, *scale_divisor=1*, *resize_type='Resize'*, ***resize_kwargs*)

First rescale the image for its shorter side to reach the short_size and then jitter its aspect ratio, final rescale the shape guaranteed to be divided by scale_divisor.

Required Keys:

- img

- img_shape
- gt_bboxes (optional)
- gt_polygons (optional)

Modified Keys:

- img
- img_shape
- gt_bboxes (optional)
- gt_polygons (optional)

Added Keys:

- scale
- scale_factor
- keep_ratio

参数

- **short_size** (*int*) –Target shorter size before jittering the aspect ratio. Defaults to 736.
- **short_size_jitter_range** (*tuple(float, float)*) –Range of the ratio used to jitter the target shorter size. Defaults to (0.7, 1.3).
- **aspect_ratio_jitter_range** (*tuple(float, float)*) –Range of the ratio used to jitter its aspect ratio. Defaults to (0.9, 1.1).
- **scale_divisor** (*int*) –The scale divisor. Defaults to 1.
- **resize_type** (*str*) –The type of resize class to use. Defaults to "Resize".
- ****resize_kwargs** –Other keyword arguments for the `resize_type`.
- **ratio_range** (*Tuple[float, float]*) –
- **aspect_ratio_range** (*Tuple[float, float]*) –

返回类型 None

transform(*results*)

Short Scale Aspect Jitter. :param results: Result dict containing the data to transform. :type results: dict

返回 The transformed data.

返回类型 dict

参数 **results** (*Dict*) –

class mmocr.datasets.transforms.**SourceImagePad**(*target_scale*, *crop_ratio=0.1111111111111111*)

Pad Image to target size. It will randomly crop an area from the original image and resize it to the target size, then paste the original image to its top left corner.

Required Keys:

- img

Modified Keys:

- img
- img_shape

Added Keys: - pad_shape - pad_fixed_size

参数

- **target_scale** (*int or tuple[int, int]*) –The target size of padded image. If it's an integer, then the padding size would be (target_size, target_size). If it's tuple, then `target_scale[0]` should be the width and `target_scale[1]` should be the height. The size of the padded image will be (target_scale[1], target_scale[0])
- **crop_ratio** (*float or Tuple[float, float]*) –Relative size for the crop region. If `crop_ratio` is a float, then the initial crop size would be `(crop_ratio * img.shape[0], crop_ratio * img.shape[1])`. If `crop_ratio` is a tuple, then `crop_ratio[0]` is for the width and `crop_ratio[1]` is for the height. The initial crop size would be `(crop_ratio[1] * img.shape[0], crop_ratio[0] * img.shape[1])`. Defaults to 1./9.

返回类型 None

transform(*results*)

Pad Image to target size. It will randomly select a small area from the original image and resize it to the target size, then paste the original image to its top left corner.

参数 **results** (*Dict*) –Result dict containing the data to transform.

返回 The transformed data.

返回类型 (Dict)

class mmocr.datasets.transforms.**TextDetRandomCrop**(*target_size*, *positive_sample_ratio=0.625*)

Randomly select a region and crop images to a target size and make sure to contain text region. This transform may break up text instances, and for broken text instances, we will crop it's bbox and polygon coordinates. This transform is recommend to be used in segmentation-based network.

Required Keys:

- img
- gt_polygons

- gt_bboxes
- gt_bboxes_labels
- gt_ignored

Modified Keys:

- img
- img_shape
- gt_polygons
- gt_bboxes
- gt_bboxes_labels
- gt_ignored

参数

- **target_size** (*tuple(int, int) or int*) –Target size for the cropped image. If it' s a tuple, then target width and target height will be `target_size[0]` and `target_size[1]`, respectively. If it' s an integer, them both target width and target height will be `target_size`.
- **positive_sample_ratio** (*float*) –The probability of sampling regions that go through text regions. Defaults to 5. / 8.

返回类型 None

transform(*results*)

Applying random crop on results. :param results: Result dict contains the data to transform. :type results: dict

返回 The transformed data

返回类型 dict

参数 **results** (*Dict*) –

class mmocr.datasets.transforms.**TextDetRandomCropFlip**(*pad_ratio=0.1*, *crop_ratio=0.5*, *iter_num=1*, *min_area_ratio=0.2*, *epsilon=0.01*)

Random crop and flip a patch in the image. Only used in text detection task.

Required Keys:

- img
- gt_bboxes
- gt_polygons

Modified Keys:

- img
- gt_bboxes
- gt_polygons

参数

- **pad_ratio** (*float*) –The ratio of padding. Defaults to 0.1.
- **crop_ratio** (*float*) –The ratio of cropping. Defaults to 0.5.
- **iter_num** (*int*) –Number of operations. Defaults to 1.
- **min_area_ratio** (*float*) –Minimal area ratio between cropped patch and original image. Defaults to 0.2.
- **epsilon** (*float*) –The threshold of polygon IoU between cropped area and polygon, which is used to avoid cropping text instances. Defaults to 0.01.

返回类型 None

transform(*results*)

Applying random crop flip on results.

参数 **results** (*dict*) –Result dict containing the data to transform

返回 The transformed data

返回类型 dict

class mmocr.datasets.transforms.**TorchVisionWrapper**(*op*, ***kwargs*)

A wrapper around torchvision trasnforms. It applies specific transform to `img` and updates `height` and `width` accordingly.

Required Keys:

- img (ndarray): The input image.

Modified Keys:

- img (ndarray): The modified image.
- img_shape (tuple(int, int)): The shape of the image in (height, width).

警告: This transform only affects the image but not its associated annotations, such as word bounding boxes and polygons. Therefore, it may only be applicable to text recognition tasks.

参数

- **op** (*str*) –The name of any transform class in `torchvision.transforms()`.

- ****kwargs** –Arguments that will be passed to initializer of torchvision transform.

返回类型 None

transform(*results*)

Transform the image.

参数 **results** (*dict*) –Result dict from the data loader.

返回 Transformed results.

返回类型 dict

CHAPTER 37

mmocr.engine

37.1 Hooks

class mmocr.engine.hooks.**VisualizationHook**(*enable=False*, *interval=50*, *score_thr=0.3*, *show=False*, *draw_pred=False*, *draw_gt=False*, *wait_time=0.0*, *file_client_args={'backend': 'disk'}*)

Detection Visualization Hook. Used to visualize validation and testing process prediction results.

参数

- **enable** (*bool*) –Whether to enable this hook. Defaults to False.
- **interval** (*int*) –The interval of visualization. Defaults to 50.
- **score_thr** (*float*) –The threshold to visualize the bboxes and masks. It's only useful for text detection. Defaults to 0.3.
- **show** (*bool*) –Whether to display the drawn image. Defaults to False.
- **wait_time** (*float*) –The interval of show in seconds. Defaults to 0.
- **file_client_args** (*dict*) –Arguments to instantiate a FileClient. See mmengine.fileio.FileClient for details. Defaults to dict(backend='disk').
- **draw_pred** (*bool*) –
- **draw_gt** (*bool*) –

返回类型 None

after_test_iter(*runner*, *batch_idx*, *data_batch*, *outputs*)

Run after every testing iterations.

参数

- **runner** (Runner) –The runner of the testing process.
- **batch_idx** (*int*) –The index of the current batch in the val loop.
- **data_batch** (*Sequence[dict]*) –Data from dataloader.
- **outputs** (*Sequence[Union[mmocr.structures.textdet_data_sample.TextDetDataSample, mmocr.structures.textrecog_data_sample.TextRecogDataSample]]*) –

返回类型 None

:param outputs (Sequence[TextDetDataSample or: TextRecogDataSample]): Outputs from model.

after_val_iter(*runner*, *batch_idx*, *data_batch*, *outputs*)

Run after every `self.interval` validation iterations.

参数

- **runner** (Runner) –The runner of the validation process.
- **batch_idx** (*int*) –The index of the current batch in the val loop.
- **data_batch** (*Sequence[dict]*) –Data from dataloader.
- **outputs** (*Sequence[Union[mmocr.structures.textdet_data_sample.TextDetDataSample, mmocr.structures.textrecog_data_sample.TextRecogDataSample]]*) –

返回类型 None

:param outputs (Sequence[TextDetDataSample or: TextRecogDataSample]): Outputs from model.

CHAPTER 38

mmocr.evaluation

38.1 Evaluator

class mmocr.evaluation.evaluator.**MultiDatasetsEvaluator**(*metrics*, *dataset_prefixes*)

Wrapper class to compose class: *ConcatDataset* and multiple `BaseMetric` instances. The metrics will be evaluated on each dataset slice separately. The name of the each metric is the concatenation of the dataset prefix, the metric prefix and the key of metric - e.g. *dataset_prefix/metric_prefix/accuracy*.

参数

- **metrics** (*dict or BaseMetric or Sequence*) –The config of metrics.
- **dataset_prefixes** (*Sequence[str]*) –The prefix of each dataset. The length of this sequence should be the same as the length of the datasets.

返回类型 None

evaluate(*size*)

Invoke `evaluate` method of each metric and collect the metrics dictionary.

参数 **size** (*int*) –Length of the entire validation dataset. When batch size > 1, the dataloader may pad some data samples to make sure all ranks have the same length of dataset slice. The `collect_results` function will drop the padded data based on this size.

返回 Evaluation results of all metrics. The keys are the names of the metrics, and the values are corresponding results.

返回类型 dict

38.2 Functional

mmocr.evaluation.functional.**compute_hmean**(*accum_hit_recall*, *accum_hit_prec*, *gt_num*, *pred_num*)

Compute hmean given hit number, ground truth number and prediction number.

参数

- **accum_hit_recall** (*int | float*) –Accumulated hits for computing recall.
- **accum_hit_prec** (*int | float*) –Accumulated hits for computing precision.
- **gt_num** (*int*) –Ground truth number.
- **pred_num** (*int*) –Prediction number.

返回 The recall value. precision (float): The precision value. hmean (float): The hmean value.

返回类型 recall (float)

38.3 Metric

class mmocr.evaluation.metrics.**CharMetric**(*valid_symbol='[^A-Z^a-z^0-9^ —-🄵]'*, *collect_device='cpu'*, *prefix=None*)

Character metrics for text recognition task.

参数

- **valid_symbol** (*str*) –Valid characters. Defaults to '[^A-Z^a-z^0-9^ —-🄵]'
- **collect_device** (*str*) –Device name used for collecting results from different ranks during distributed training. Must be 'cpu' or 'gpu'. Defaults to 'cpu'.
- **prefix** (*str, optional*) –The prefix that will be added in the metric names to disambiguate homonymous metrics of different evaluators. If prefix is not provided in the argument, self.default_prefix will be used instead. Defaults to None.

返回类型 None

compute_metrics(*results*)

Compute the metrics from processed results.

参数 **results** (*list[Dict]*) –The processed results of each batch.

返回 The computed metrics. The keys are the names of the metrics, and the values are corresponding results.

返回类型 Dict

process(*data_batch*, *data_samples*)

Process one batch of data_samples. The processed results should be stored in `self.results`, which will be used to compute the metrics when all batches have been processed.

参数

- **data_batch** (*Sequence[Dict]*) –A batch of gts.
- **data_samples** (*Sequence[Dict]*) –A batch of outputs from the model.

返回类型 None

class mmocr.evaluation.metrics.**F1Metric**(*num_classes*, *key='labels'*, *mode='micro'*, *cared_classes=[]*, *ignored_classes=[]*, *collect_device='cpu'*, *prefix=None*)

Compute F1 scores.

参数

- **num_classes** (*int*) –Number of labels.
- **key** (*str*) –The key name of the predicted and ground truth labels. Defaults to 'labels'.
- **mode** (*str or list[str]*) –Options are: - 'micro': Calculate metrics globally by counting the total true

  positives, false negatives and false positives.

  – 'macro': Calculate metrics for each label, and find their unweighted mean.

  If mode is a list, then metrics in mode will be calculated separately. Defaults to 'micro'.
- **cared_classes** (*list[int]*) –The indices of the labels particpated in the metirc computing. If both `cared_classes` and `ignored_classes` are empty, all classes will be taken into account. Defaults to []. Note: `cared_classes` and `ignored_classes` cannot be specified together.
- **ignored_classes** (*list[int]*) –The index set of labels that are ignored when computing metrics. If both `cared_classes` and `ignored_classes` are empty, all classes will be taken into account. Defaults to []. Note: `cared_classes` and `ignored_classes` cannot be specified together.
- **collect_device** (*str*) –Device name used for collecting results from different ranks during distributed training. Must be 'cpu' or 'gpu'. Defaults to 'cpu'.
- **prefix** (*str, optional*) –The prefix that will be added in the metric names to disambiguate homonymous metrics of different evaluators. If prefix is not provided in the argument, self.default_prefix will be used instead. Defaults to None.

返回类型 None

> **警告：** Only non-negative integer labels are involved in computing. All negative ground truth labels will be ignored.

compute_metrics(*results*)

Compute the metrics from processed results.

参数 **results** (*list[Dict]*) –The processed results of each batch.

返回

The f1 scores. The keys are the names of the metrics, and the values are corresponding results. Possible keys are 'micro_f1' and 'macro_f1'.

返回类型 dict[str, float]

process(*data_batch*, *data_samples*)

Process one batch of data_samples. The processed results should be stored in `self.results`, which will be used to compute the metrics when all batches have been processed.

参数

- **data_batch** (*Sequence[Dict]*) –A batch of gts.
- **data_samples** (*Sequence[Dict]*) –A batch of outputs from the model.

返回类型 None

class mmocr.evaluation.metrics.**HmeanIOUMetric**(*match_iou_thr=0.5*, *ignore_precision_thr=0.5*, *pred_score_thrs={'start': 0.3, 'step': 0.1, 'stop': 0.9}*, *strategy='vanilla'*, *collect_device='cpu'*, *prefix=None*)

HmeanIOU metric.

This method computes the hmean iou metric, which is done in the following steps:

- Filter the prediction polygon:
  - Scores is smaller than minimum prediction score threshold.
  - The proportion of the area that intersects with gt ignored polygon is greater than ignore_precision_thr.
- Computing an M x N IoU matrix, where each element indexing E_mn represents the IoU between the m-th valid GT and n-th valid prediction.
- Based on different prediction score threshold: - Obtain the ignored predictions according to prediction score.

  The filtered predictions will not be involved in the later metric computations.

  - Based on the IoU matrix, get the match metric according to

`match_iou_thr`. - Based on different *strategy*, accumulate the match number.

- calculate H-mean under different prediction score threshold.

参数

- **match_iou_thr** (*float*) –IoU threshold for a match. Defaults to 0.5.
- **ignore_precision_thr** (*float*) –Precision threshold when prediction and gt ignored polygons are matched. Defaults to 0.5.
- **pred_score_thrs** (*dict*) –Best prediction score threshold searching space. Defaults to dict(start=0.3, stop=0.9, step=0.1).
- **strategy** (*str*) –Polygon matching strategy. Options are ‘max_matching’ and ‘vanilla’. ‘max_matching’ refers to the optimum strategy that maximizes the number of matches. Vanilla strategy matches gt and pred polygons if both of them are never matched before. It was used in MMOCR 0.x and and academia. Defaults to ‘vanilla’.
- **collect_device** (*str*) –Device name used for collecting results from different ranks during distributed training. Must be ‘cpu’ or ‘gpu’. Defaults to ‘cpu’.
- **prefix** (*str, optional*) –The prefix that will be added in the metric names to disambiguate homonymous metrics of different evaluators. If prefix is not provided in the argument, self.default_prefix will be used instead. Defaults to None

返回类型 None

compute_metrics(*results*)

Compute the metrics from processed results.

参数 **results** (*list[dict]*) –The processed results of each batch.

返回 The computed metrics. The keys are the names of the metrics, and the values are corresponding results.

返回类型 dict

process(*data_batch*, *data_samples*)

Process one batch of data samples and predictions. The processed results should be stored in `self.results`, which will be used to compute the metrics when all batches have been processed.

参数

- **data_batch** (*Sequence[Dict]*) –A batch of data from dataloader.
- **data_samples** (*Sequence[Dict]*) –A batch of outputs from the model.

返回类型 None

class mmocr.evaluation.metrics.**OneMinusNEDMetric**(*valid_symbol='[^A-Z^a-z^0-9^ 一-🄵]'*, *collect_device='cpu'*, *prefix=None*)

One minus NED metric for text recognition task.

参数

- **valid_symbol** (*str*) –Valid characters. Defaults to '[^A-Z^a-z^0-9^ —-F]'
- **collect_device** (*str*) –Device name used for collecting results from different ranks during distributed training. Must be 'cpu' or 'gpu'. Defaults to 'cpu'.
- **prefix** (*str, optional*) –The prefix that will be added in the metric names to disambiguate homonymous metrics of different evaluators. If prefix is not provided in the argument, self.default_prefix will be used instead. Defaults to None

返回类型 None

compute_metrics(*results*)

Compute the metrics from processed results.

参数 **results** (*list[Dict]*) –The processed results of each batch.

返回 The computed metrics. The keys are the names of the metrics, and the values are corresponding results.

返回类型 Dict

process(*data_batch*, *data_samples*)

Process one batch of data_samples. The processed results should be stored in `self.results`, which will be used to compute the metrics when all batches have been processed.

参数

- **data_batch** (*Sequence[Dict]*) –A batch of gts.
- **data_samples** (*Sequence[Dict]*) –A batch of outputs from the model.

返回类型 None

class mmocr.evaluation.metrics.**WordMetric**(*mode='ignore_case_symbol'*, *valid_symbol='[^A-Z^a-z^0-9^ —-F]'*, *collect_device='cpu'*, *prefix=None*)

Word metrics for text recognition task.

参数

- **mode** (*str or list[str]*) –Options are: - 'exact': Accuracy at word level. - 'ignore_case': Accuracy at word level, ignoring letter

  case.

  – 'ignore_case_symbol': Accuracy at word level, ignoring letter case and symbol. (Default metric for academic evaluation)

  If mode is a list, then metrics in mode will be calculated separately. Defaults to 'ignore_case_symbol'

- **valid_symbol** (*str*) –Valid characters. Defaults to '[^A-Z^a-z^0-9^ —-F]'
- **collect_device** (*str*) –Device name used for collecting results from different ranks during distributed training. Must be 'cpu' or 'gpu'. Defaults to 'cpu'.
- **prefix** (*str, optional*) –The prefix that will be added in the metric names to disambiguate homonymous metrics of different evaluators. If prefix is not provided in the argument, self.default_prefix will be used instead. Defaults to None.

返回类型 None

compute_metrics(*results*)

Compute the metrics from processed results.

参数 **results** (*list[Dict]*) –The processed results of each batch.

返回 The computed metrics. The keys are the names of the metrics, and the values are corresponding results.

返回类型 Dict

process(*data_batch*, *data_samples*)

Process one batch of data_samples. The processed results should be stored in `self.results`, which will be used to compute the metrics when all batches have been processed.

参数

- **data_batch** (*Sequence[Dict]*) –A batch of gts.
- **data_samples** (*Sequence[Dict]*) –A batch of outputs from the model.

返回类型 None

CHAPTER 39

mmocr.utils

39.1 Point utils

mmocr.utils.point_utils.**point_distance**(*pt1*, *pt2*)

Calculate the distance between two points.

参数

- **pt1** (*ArrayLike*) –The first point.
- **pt2** (*ArrayLike*) –The second point.

返回 The distance between two points.

返回类型 float

mmocr.utils.point_utils.**points_center**(*points*)

Calculate the center of a set of points.

参数 **points** (*ArrayLike*) –A set of points.

返回 The coordinate of center point.

返回类型 np.ndarray

39.2 Bbox utils

mmocr.utils.bbox_utils.**bbox2poly**(*bbox*)

Converting a bounding box to a polygon.

参数 **bbox** (*ArrayLike*) –A bbox. In any form can be accessed by 1-D indices. E.g. list[float], np.ndarray, or torch.Tensor. bbox is written in

[x1, y1, x2, y2].

返回 The converted polygon [x1, y1, x2, y1, x2, y2, x1, y2].

返回类型 np.array

mmocr.utils.bbox_utils.**bbox_center_distance**(*box1*, *box2*)

Calculate the distance between the center points of two bounding boxes.

参数

- **box1** (*ArrayLike*) –The first bounding box represented in [x1, y1, x2, y2].
- **box2** (*ArrayLike*) –The second bounding box represented in [x1, y1, x2, y2].

返回 The distance between the center points of two bounding boxes.

返回类型 float

mmocr.utils.bbox_utils.**bbox_diag_distance**(*box*)

Calculate the diagonal length of a bounding box (distance between the top-left and bottom-right).

参数

- **box** (*ArrayLike*) –The bounding box represented in
- **[x1** –
- **y1** –
- **x2** –
- **y2** –
- **x3** –
- **y3** –
- **x4** –
- **or [x1** (*y4]*) –
- **y1** –
- **x2** –
- **y2]** –

返回 The diagonal length of the bounding box.

返回类型 float

mmocr.utils.bbox_utils.**bbox_jitter**(*points_x*, *points_y*, *jitter_ratio_x=0.5*, *jitter_ratio_y=0.1*)

Jitter on the coordinates of bounding box.

参数

- **points_x** (*list[float | int]*) –List of y for four vertices.
- **points_y** (*list[float | int]*) –List of x for four vertices.
- **jitter_ratio_x** (*float*) –Horizontal jitter ratio relative to the height.
- **jitter_ratio_y** (*float*) –Vertical jitter ratio relative to the height.

mmocr.utils.bbox_utils.**bezier2polygon**(*bezier_points*, *num_sample=20*)

Sample points from the boundary of a polygon enclosed by two Bezier curves, which are controlled by bezier_points.

参数

- **bezier_points** (*ndarray*) –A $(2, 4, 2)$ array of 8 Bezeir points or its equalivance. The first 4 points control the curve at one side and the last four control the other side.
- **num_sample** (*int*) –The number of sample points at each Bezeir curve. Defaults to 20.

返回 A list of 2*num_sample points representing the polygon extracted from Bezier curves.

返回类型 list[ndarray]

警告: The points are not guaranteed to be ordered. Please use mmocr.utils.sort_points() to sort points if necessary.

mmocr.utils.bbox_utils.**is_on_same_line**(*box_a*, *box_b*, *min_y_overlap_ratio=0.8*)

Check if two boxes are on the same line by their y-axis coordinates.

Two boxes are on the same line if they overlap vertically, and the length of the overlapping line segment is greater than min_y_overlap_ratio * the height of either of the boxes.

参数

- **box_a** (*list), box_b (list*) –Two bounding boxes to be checked
- **min_y_overlap_ratio** (*float*) –The minimum vertical overlapping ratio allowed for boxes in the same line

返回 The bool flag indicating if they are on the same line

mmocr.utils.bbox_utils.**rescale_bbox**(*bbox*, *scale_factor*, *mode='mul'*)

Rescale a bounding box according to scale_factor.

The behavior is different depending on the mode. When mode is ‘mul’, the coordinates will be multiplied by scale_factor, which is usually used in preprocessing transforms such as `Resize()`. The coordinates will be divided by scale_factor if mode is ‘div’. It can be used in postprocessors to recover the bbox in the original image size.

参数

- **bbox** (*ndarray*) –A bounding box [x1, y1, x2, y2].
- **scale_factor** (*tuple(int, int)*) –(w_scale, h_scale).
- **model** (*str*) –Rescale mode. Can be ‘mul’ or ‘div’. Defaults to ‘mul’.
- **mode** (*str*) –

返回 Rescaled bbox.

返回类型 np.ndarray

mmocr.utils.bbox_utils.**rescale_bboxes**(*bboxes*, *scale_factor*, *mode='mul'*)

Rescale bboxes according to scale_factor.

The behavior is different depending on the mode. When mode is ‘mul’, the coordinates will be multiplied by scale_factor, which is usually used in preprocessing transforms such as `Resize()`. The coordinates will be divided by scale_factor if mode is ‘div’. It can be used in postprocessors to recover the bboxes in the original image size.

参数

- **bboxes** (*np.ndarray]*) –Bounding bboxes in shape (N, 4)
- **scale_factor** (*tuple(int, int)*) –(w_scale, h_scale).
- **model** (*str*) –Rescale mode. Can be ‘mul’ or ‘div’. Defaults to ‘mul’.
- **mode** (*str*) –

返回 Rescaled bboxes.

返回类型 list[np.ndarray]

mmocr.utils.bbox_utils.**sort_points**(*points*)

Sort arbitory points in clockwise order. Reference: https://stackoverflow.com/a/6989383.

参数 **points** (*list[ndarray] or ndarray or list[list]*) –A list of unsorted boundary points.

返回 A list of points sorted in clockwise order.

返回类型 list[ndarray]

mmocr.utils.bbox_utils.**sort_vertex**(*points_x*, *points_y*)

Sort box vertices in clockwise order from left-top first.

参数

- **points_x** (*list[float]*) –x of four vertices.
- **points_y** (*list[float]*) –y of four vertices.

返回 x of sorted four vertices. sorted_points_y (list[float]): y of sorted four vertices.

返回类型 sorted_points_x (list[float])

mmocr.utils.bbox_utils.**sort_vertex8**(*points*)

Sort vertex with 8 points [x1 y1 x2 y2 x3 y3 x4 y4]

mmocr.utils.bbox_utils.**stitch_boxes_into_lines**(*boxes*, *max_x_dist=10*, *min_y_overlap_ratio=0.8*)

Stitch fragmented boxes of words into lines.

Note: part of its logic is inspired by @Johndirr (https://github.com/faustomorales/keras-ocr/issues/22)

参数

- **boxes** (*list*) –List of ocr results to be stitched
- **max_x_dist** (*int*) –The maximum horizontal distance between the closest edges of neighboring boxes in the same line
- **min_y_overlap_ratio** (*float*) –The minimum vertical overlapping ratio allowed for any pairs of neighboring boxes in the same line

返回 List of merged boxes and texts

返回类型 merged_boxes(list[dict])

39.3 Polygon utils

mmocr.utils.polygon_utils.**boundary_iou**(*src*, *target*, *zero_division=0*)

Calculate the IOU between two boundaries.

参数

- **src** (*list*) –Source boundary.
- **target** (*list*) –Target boundary.
- **zero_division** (*int or float*) –The return value when invalid boundary exists.

返回 The iou between two boundaries.

返回类型 float

mmocr.utils.polygon_utils.**crop_polygon**(*polygon*, *crop_box*)

Crop polygon to be within a box region.

参数

- **polygon** (*ndarray*) –polygon in shape (N,).
- **crop_box** (*ndarray*) –target box region in shape (4,).

返回

Cropped polygon. If the polygon is not within the crop box, return None.

返回类型 np.array or None

mmocr.utils.polygon_utils.**is_poly_inside_rect**(*poly*, *rect*)

Check if the polygon is inside the target region. :param poly: Polygon in shape (N,). :type poly: ArrayLike :param rect: Target region [x1, y1, x2, y2]. :type rect: ndarray

返回 Whether the polygon is inside the cropping region.

返回类型 bool

参数

- **poly**(*Union[Sequence[Sequence[Sequence[Sequence[Sequence[Any]]]]], numpy.typing._array_like._SupportsArray[numpy.dtype], Sequence[numpy.typing._array_like._SupportsArray[numpy.dtype]], Sequence[Sequence[numpy.typing._array_like._SupportsArray[numpy.dtype]]], Sequence[Sequence[Sequence[numpy.typing._array_like._SupportsArray[numpy.dtype]]]], Sequence[Sequence[Sequence[Sequence[numpy.typing._array_like._SupportsArray[numpy.dtype]]]]], bool, int, float, complex, str, bytes, Sequence[Union[bool, int, float, complex, str, bytes]], Sequence[Sequence[Union[bool, int, float, complex, str, bytes]]], Sequence[Sequence[Sequence[Union[bool, int, float, complex, str, bytes]]]], Sequence[Sequence[Sequence[Sequence[Union[bool, int, float, complex, str, bytes]]]]]]*) –
- **rect** (*numpy.ndarray*) –

mmocr.utils.polygon_utils.**offset_polygon**(*poly*, *distance*)

Offset (expand/shrink) the polygon by the target distance. It's a wrapper around pyclipper based on Vatti clipping algorithm.

> **警告:** Polygon coordinates will be casted to int type in PyClipper. Mind the potential precision loss caused by the casting.

参数

- **poly** (*ArrayLike*) –A polygon. In any form can be converted to an 1-D numpy array. E.g. list[float], np.ndarray, or torch.Tensor. Polygon is written in [x1, y1, x2, y2, ···].
- **distance** (*float*) –The offset distance. Positive value means expanding, negative value means shrinking.

返回 1-D Offsetted polygon ndarray in float32 type. If the result polygon is invalid or has been split into several parts, return an empty array.

返回类型 np.array

mmocr.utils.polygon_utils.**poly2bbox**(*polygon*)

Converting a polygon to a bounding box.

参数 **polygon** –A polygon. In any form can be converted to an 1-D numpy array. E.g. list[float], np.ndarray, or torch.Tensor. Polygon is written in [x1, y1, x2, y2, ···].

返回类型 numpy.array

mmocr.utils.polygon_utils.**poly2shapely**(*polygon*)

Convert a polygon to shapely.geometry.Polygon.

参数 **polygon** (*ArrayLike*) –A set of points of 2k shape.

返回 A polygon object.

返回类型 polygon (Polygon)

mmocr.utils.polygon_utils.**poly_intersection**(*poly_a*, *poly_b*, *invalid_ret=None*, *return_poly=False*)

Calculate the intersection area between two polygons.

参数

- **poly_a** (*Polygon*) –Polygon a.
- **poly_b** (*Polygon*) –Polygon b.
- **invalid_ret** (*float or int, optional*) –The return value when the invalid polygon exists. If it is not specified, the function allows the computation to proceed with invalid polygons by cleaning the their self-touching or self-crossing parts. Defaults to None.
- **return_poly** (*bool*) –Whether to return the polygon of the intersection Defaults to False.

返回 Returns the intersection area or a tuple `(area, Optional[poly_obj])`, where the *area* is the intersection area between two polygons and *poly_obj* is The Polygon object of the intersection area. Set as *None* if the input is invalid. Set as *None* if the input is invalid. *poly_obj* will be returned only if *return_poly* is *True*.

返回类型 float or tuple(float, Polygon)

mmocr.utils.polygon_utils.**poly_iou**(*poly_a*, *poly_b*, *zero_division=0.0*)

Calculate the IOU between two polygons.

参数

- **poly_a** (*Polygon*) –Polygon a.
- **poly_b** (*Polygon*) –Polygon b.
- **zero_division** (*float*) –The return value when invalid polygon exists.

返回 The IoU between two polygons.

返回类型 float

mmocr.utils.polygon_utils.**poly_make_valid**(*poly*)

Convert a potentially invalid polygon to a valid one by eliminating self-crossing or self-touching parts.

参数 **poly** (*Polygon*) –A polygon needed to be converted.

返回 A valid polygon.

返回类型 Polygon

mmocr.utils.polygon_utils.**poly_union**(*poly_a*, *poly_b*, *invalid_ret=None*, *return_poly=False*)

Calculate the union area between two polygons. :param poly_a: Polygon a. :type poly_a: Polygon :param poly_b: Polygon b. :type poly_b: Polygon :param invalid_ret: The return value when the

invalid polygon exists. If it is not specified, the function allows the computation to proceed with invalid polygons by cleaning the their self-touching or self-crossing parts. Defaults to False.

参数

- **return_poly** (*bool*) –Whether to return the polygon of the union. Defaults to False.
- **poly_a** (*shapely.geometry.polygon.Polygon*) –
- **poly_b** (*shapely.geometry.polygon.Polygon*) –
- **invalid_ret** (*float or int, optional*) –

返回 Returns a tuple `(area, Optional[poly_obj])`, where the *area* is the union between two polygons and *poly_obj* is the Polygon or MultiPolygon object of the union of the inputs. The type of object depends on whether they intersect or not. Set as *None* if the input is invalid. *poly_obj* will be returned only if *return_poly* is *True*.

返回类型 tuple

mmocr.utils.polygon_utils.**polys2shapely**(*polygons*)

Convert a nested list of boundaries to a list of Polygons.

参数 **polygons** (*list*) –The point coordinates of the instance boundary.

返回 Converted shapely.Polygon.

返回类型 list

mmocr.utils.polygon_utils.**rescale_polygon**(*polygon*, *scale_factor*, *mode='mul'*)

Rescale a polygon according to scale_factor.

The behavior is different depending on the mode. When mode is 'mul', the coordinates will be multiplied by scale_factor, which is usually used in preprocessing transforms such as `Resize()`. The coordinates will be divided by scale_factor if mode is 'div'. It can be used in postprocessors to recover the polygon in the original image size.

参数

- **polygon** (*ArrayLike*) –A polygon. In any form can be converted to an 1-D numpy array. E.g. list[float], np.ndarray, or torch.Tensor. Polygon is written in [x1, y1, x2, y2, ···].
- **scale_factor** (*tuple(int, int)*) –(w_scale, h_scale).
- **model** (*str*) –Rescale mode. Can be 'mul' or 'div'. Defaults to 'mul'.
- **mode** (*str*) –

返回 Rescaled polygon.

返回类型 np.ndarray

mmocr.utils.polygon_utils.**rescale_polygons**(*polygons*, *scale_factor*, *mode='mul'*)

Rescale polygons according to scale_factor.

The behavior is different depending on the mode. When mode is 'mul', the coordinates will be multiplied by scale_factor, which is usually used in preprocessing transforms such as `Resize()`. The coordinates will be divided by scale_factor if mode is 'div'. It can be used in postprocessors to recover the polygon in the original image size.

参数

- **polygons** (*list[ArrayLike]*) –A list of polygons, each written in [x1, y1, x2, y2, ···] and in any form can be converted to an 1-D numpy array. E.g. list[list[float]], list[np.ndarray], or list[torch.Tensor].
- **scale_factor** (*tuple(int, int)*) –(w_scale, h_scale).
- **model** (*str*) –Rescale mode. Can be 'mul' or 'div'. Defaults to 'mul'.
- **mode** (*str*) –

返回 Rescaled polygons.

返回类型 list[np.ndarray]

mmocr.utils.polygon_utils.**shapely2poly**(*polygon*)

Convert a nested list of boundaries to a list of Polygons.

参数 **polygon** (*Polygon*) –A polygon represented by shapely.Polygon.

返回 Converted numpy array

返回类型 np.array

mmocr.utils.polygon_utils.**sort_points**(*points*)

Sort arbitory points in clockwise order. Reference: https://stackoverflow.com/a/6989383.

参数 **points** (*list[ndarray] or ndarray or list[list]*) –A list of unsorted boundary points.

返回 A list of points sorted in clockwise order.

返回类型 list[ndarray]

mmocr.utils.polygon_utils.**sort_vertex**(*points_x*, *points_y*)

Sort box vertices in clockwise order from left-top first.

参数

- **points_x** (*list[float]*) –x of four vertices.
- **points_y** (*list[float]*) –y of four vertices.

返回

Sorted x and y of four vertices.

- sorted_points_x (list[float]): x of sorted four vertices.
- sorted_points_y (list[float]): y of sorted four vertices.

返回类型 tuple[list[float], list[float]]

mmocr.utils.polygon_utils.**sort_vertex8**(*points*)

Sort vertex with 8 points [x1 y1 x2 y2 x3 y3 x4 y4]

39.4 Mask utils

mmocr.utils.mask_utils.**fill_hole**(*input_mask*)

Fill holes in matrix.

Input:

[[0, 0, 0, 0, 0, 0, 0], [0, 1, 1, 1, 1, 1, 0], [0, 1, 0, 0, 0, 1, 0], [0, 1, 1, 1, 1, 1, 0], [0, 0, 0, 0, 0, 0, 0]]

Output:

[[0, 0, 0, 0, 0, 0, 0], [0, 1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 1, 0], [0, 0, 0, 0, 0, 0, 0]]

参数 **input_mask** (*ArrayLike*) –The input mask.

返回 The output mask that has been filled.

返回类型 np.array

39.5 String utils

class mmocr.utils.string_utils.**StringStripper**(*strip=True*, *strip_pos='both'*, *strip_str=None*)

Removing the leading and/or the trailing characters based on the string argument passed.

参数

- **strip** (*bool*) –Whether remove characters from both left and right of the string. Default: True.
- **strip_pos** (*str*) –Which position for removing, can be one of ('both', 'left', 'right'), Default: 'both'.
- **strip_str** (*str|None*) –A string specifying the set of characters to be removed from the left and right part of the string. If None, all leading and trailing whitespaces are removed from the string. Default: None.

39.6 Image utils

mmocr.utils.img_utils.**crop_img**(*src_img*, *box*, *long_edge_pad_ratio=0.4*, *short_edge_pad_ratio=0.2*)

Crop text region given the bounding box which might be slightly padded. The bounding box is assumed to be a quadrangle and tightly bound the text region.

参数

- **src_img** (*np.array*) –The original image.
- **box** (*list[float | int]*) –Points of quadrangle.
- **long_edge_pad_ratio** (*float*) –The ratio of padding to the long edge. The padding will be the length of the short edge * long_edge_pad_ratio. Defaults to 0.4.
- **short_edge_pad_ratio** (*float*) –The ratio of padding to the short edge. The padding will be the length of the long edge * short_edge_pad_ratio. Defaults to 0.2.

返回 The cropped image.

返回类型 np.array

mmocr.utils.img_utils.**warp_img**(*src_img*, *box*, *jitter=False*, *jitter_ratio_x=0.5*, *jitter_ratio_y=0.1*)

Crop box area from image using opencv warpPerspective.

参数

- **src_img** (*np.array*) –Image before cropping.
- **box** (*list[float | int]*) –Coordinates of quadrangle.
- **jitter** (*bool*) –Whether to jitter the box.

- **jitter_ratio_x** (*float*) –Horizontal jitter ratio relative to the height.
- **jitter_ratio_y** (*float*) –Vertical jitter ratio relative to the height.

返回 The warped image.

返回类型 np.array

39.7 File IO utils

mmocr.utils.fileio.**list_from_file**(*filename*, *encoding='utf-8'*)

Load a text file and parse the content as a list of strings. The trailing "r" and "n" of each line will be removed.

注解: This will be replaced by mmcv's version after it supports encoding.

参数

- **filename** (*str*) –Filename.
- **encoding** (*str*) –Encoding used to open the file. Default utf-8.

返回 A list of strings.

返回类型 list[str]

mmocr.utils.fileio.**list_to_file**(*filename*, *lines*)

Write a list of strings to a text file.

参数

- **filename** (*str*) –The output filename. It will be created/overwritten.
- **lines** (*list(str)*) –Data to be written.

39.8 Others

mmocr.utils.data_converter_utils.**dump_ocr_data**(*image_infos*, *out_json_name*, *task_name*, ***kwargs*)

Dump the annotation in openmmlab style.

参数

- **image_infos** (*list*) –List of image information dicts. Read the example section for the format illustration.
- **out_json_name** (*str*) –Output json filename.

- **task_name** (*str*) –Task name. Options are ‘textdet’, ‘textrecog’ and ‘textspotter’.

返回类型 Dict

实际案例

Here is the general structure of image_infos for textdet/textspotter tasks:

```
[  # A list of dicts. Each dict stands for a single image.
    {
        "file_name": "1.jpg",
        "height": 100,
        "width": 200,
        "segm_file": "seg.txt" # (optional) path to segmap
        "anno_info": [  # a list of dicts. Each dict
                        # stands for a single text instance.
            {
                "iscrowd": 0,  # 0: don't ignore this instance
                               # 1: ignore
                "category_id": 0,  # Instance class id. Must be 0
                                   # for OCR tasks to permanently
                                   # be mapped to 'text' category
                "bbox": [x, y, w, h],
                "segmentation": [x1, y1, x2, y2, ...],
                "text": "demo_text"  # for textspotter only.
            }
        ]
    },
]
```

The input for textrecog task is much simpler:

```
[   # A list of dicts. Each dict stands for a single image.
    {
        "file_name": "1.jpg",
        "anno_info": [  # a list of dicts. Each dict
                        # stands for a single text instance.
                        # However, in textrecog, usually each
                        # image only has one text instance.
            {
                "text": "demo_text"
            }
        ]
    },
]
```

返回 The openmmlab-style annotation.

返回类型 out_json(dict)

参数

- **image_infos** (*Sequence[Dict]*) –
- **out_json_name** (*str*) –
- **task_name** (*str*) –

mmocr.utils.data_converter_utils.**recog_anno_to_imginfo**(*file_paths*, *labels*)

Convert a list of file_paths and labels for recognition tasks into the format of image_infos acceptable by *dump_ocr_data()*. It's meant to maintain compatibility with the legacy annotation format in MMOCR 0.x.

In MMOCR 0.x, data converters for recognition usually converts the annotations into a list of file paths and a list of labels, which look like the following:

```
file_paths = ['1.jpg', '2.jpg', ...]
labels = ['aaa', 'bbb', ...]
```

This utility merges them into a list of dictionaries parsable by *dump_ocr_data()*:

```
[   # A list of dicts. Each dict stands for a single image.
    {
        "file_name": "1.jpg",
        "anno_info": [
            {
                "text": "aaa"
            }
        ]
    },
    {
        "file_name": "2.jpg",
        "anno_info": [
            {
                "text": "bbb"
            }
        ]
    },
    ...
]
```

参数

- **file_paths** (*list[str]*) –A list of file paths to images.

- **labels** (*list[str]*) –A list of text labels.

返回 Annotations parsable by *dump_ocr_data()*.

返回类型 list[dict]

class mmocr.utils.parsers.**LineJsonParser**(*keys=['filename', 'text']*)

Parse json-string of one line in annotation file to dict format.

参数 **keys** (*list[str]*) –Keys in both json-string and result dict. Defaults to [‘filename’, ‘text’].

返回类型 None

class mmocr.utils.parsers.**LineStrParser**(*keys=['filename', 'text']*, *keys_idx=[0, 1]*, *separator=' '*, ***kwargs*)

Parse string of one line in annotation file to dict format.

参数

- **keys** (*list[str]*) –Keys in result dict. Defaults to [‘filename’, ‘text’].
- **keys_idx** (*list[int]*) –Value index in sub-string list for each key above. Defaults to [0, 1].
- **separator** (*str*) –Separator to separate string to list of sub-string. Defaults to ‘ ‘.

CHAPTER 40

mmocr.models

40.1 Common

class mmocr.models.common.backbones.**UNet**(*in_channels=3*, *base_channels=64*, *num_stages=5*, *strides=(1, 1, 1, 1, 1)*, *enc_num_convs=(2, 2, 2, 2, 2)*, *dec_num_convs=(2, 2, 2, 2)*, *downsamples=(True, True, True, True)*, *enc_dilations=(1, 1, 1, 1, 1)*, *dec_dilations=(1, 1, 1, 1)*, *with_cp=False*, *conv_cfg=None*, *norm_cfg={'type': 'BN'}*, *act_cfg={'type': 'ReLU'}*, *upsample_cfg={'type': 'InterpConv'}*, *norm_eval=False*, *dcn=None*, *plugins=None*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Constant', 'layer': ['_BatchNorm', 'GroupNorm'], 'val': 1}]*)

UNet backbone. U-Net: Convolutional Networks for Biomedical Image Segmentation. https://arxiv.org/pdf/1505.04597.pdf

参数

- **in_channels** (*int*) –Number of input image channels. Default" 3.
- **base_channels** (*int*) –Number of base channels of each stage. The output channels of the first stage. Default: 64.
- **num_stages** (*int*) –Number of stages in encoder, normally 5. Default: 5.

- **strides** (*Sequence[int 1 | 2]*) –Strides of each stage in encoder. len(strides) is equal to num_stages. Normally the stride of the first stage in encoder is 1. If strides[i]=2, it uses stride convolution to downsample in the correspondence encoder stage. Default: (1, 1, 1, 1, 1).
- **enc_num_convs** (*Sequence[int]*) –Number of convolutional layers in the convolution block of the correspondence encoder stage. Default: (2, 2, 2, 2, 2).
- **dec_num_convs** (*Sequence[int]*) –Number of convolutional layers in the convolution block of the correspondence decoder stage. Default: (2, 2, 2, 2).
- **downsamples** (*Sequence[int]*) –Whether use MaxPool to downsample the feature map after the first stage of encoder (stages: [1, num_stages)). If the correspondence encoder stage use stride convolution (strides[i]=2), it will never use MaxPool to downsample, even downsamples[i-1]=True. Default: (True, True, True, True).
- **enc_dilations** (*Sequence[int]*) –Dilation rate of each stage in encoder. Default: (1, 1, 1, 1, 1).
- **dec_dilations** (*Sequence[int]*) –Dilation rate of each stage in decoder. Default: (1, 1, 1, 1).
- **with_cp** (*bool*) –Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed. Default: False.
- **conv_cfg** (*dict | None*) –Config dict for convolution layer. Default: None.
- **norm_cfg** (*dict | None*) –Config dict for normalization layer. Default: dict(type='BN').
- **act_cfg** (*dict | None*) –Config dict for activation layer in ConvModule. Default: dict(type='ReLU').
- **upsample_cfg** (*dict*) –The upsample config of the upsample module in decoder. Default: dict(type='InterpConv').
- **norm_eval** (*bool*) –Whether to set norm layers to eval mode, namely, freeze running stats (mean and var). Note: Effect on Batch Norm and its variants only. Default: False.
- **dcn** (*bool*) –Use deformable convolution in convolutional layer or not. Default: None.
- **plugins** (*dict*) –plugins for convolutional layers. Default: None.

Notice: The input image size should be divisible by the whole downsample rate of the encoder. More detail of the whole downsample rate can be found in UNet._check_input_divisible.

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.

---

train(*mode=True*)

Convert the model into training mode while keep normalization layer freezed.

class mmocr.models.common.losses.**CrossEntropyLoss**(*weight=None*, *size_average=None*, *ignore_index=- 100*, *reduce=None*, *reduction='mean'*, *label_smoothing=0.0*)

Cross entropy loss.

参数

- **weight** (*Optional[torch.Tensor]*) –
- **ignore_index** (*int*) –
- **reduction** (*str*) –
- **label_smoothing** (*float*) –

返回类型 None

class mmocr.models.common.losses.**MaskedBCELoss**(*eps=1e-06*)

Masked BCE loss.

参数 **eps** (*float*) –Eps to avoid zero-division error. Defaults to 1e-6.

返回类型 None

forward(*pred*, *gt*, *mask=None*)

Forward function.

参数

- **pred** (*torch.Tensor*) –The prediction in any shape.
- **gt** (*torch.Tensor*) –The learning target of the prediction in the same shape as pred.
- **mask** (*torch.Tensor, optional*) –Binary mask in the same shape of pred, indicating positive regions to calculate the loss. Whole region will be taken into account if not provided. Defaults to None.

返回 The loss value.

返回类型 torch.Tensor

class mmocr.models.common.losses.**MaskedBCEWithLogitsLoss**(*eps=1e-06*)

This loss combines a Sigmoid layers and a masked BCE loss in one single class. It' s AMP-eligible.

参数 **eps** (*float*) –Eps to avoid zero-division error. Defaults to 1e-6.

返回类型 None

forward(*pred*, *gt*, *mask=None*)

Forward function.

参数

- **pred** (*torch.Tensor*) –The prediction in any shape.
- **gt** (*torch.Tensor*) –The learning target of the prediction in the same shape as pred.
- **mask** (*torch.Tensor, optional*) –Binary mask in the same shape of pred, indicating positive regions to calculate the loss. Whole region will be taken into account if not provided. Defaults to None.

返回 The loss value.

返回类型 torch.Tensor

class mmocr.models.common.losses.**MaskedBalancedBCELoss**(*reduction='none'*, *negative_ratio=3*, *fallback_negative_num=0*, *eps=1e-06*)

Masked Balanced BCE loss.

参数

- **reduction** (*str, optional*) –The method to reduce the loss. Options are 'none', 'mean' and 'sum'. Defaults to 'none'.
- **negative_ratio** (*float or int*) –Maximum ratio of negative samples to positive ones. Defaults to 3.
- **fallback_negative_num** (*int*) –When the mask contains no positive samples, the number of negative samples to be sampled. Defaults to 0.
- **eps** (*float*) –Eps to avoid zero-division error. Defaults to 1e-6.

返回类型 None

forward(*pred*, *gt*, *mask=None*)

Forward function.

参数

- **pred** (*torch.Tensor*) –The prediction in any shape.
- **gt** (*torch.Tensor*) –The learning target of the prediction in the same shape as pred.
- **mask** (*torch.Tensor, optional*) –Binary mask in the same shape of pred, indicating positive regions to calculate the loss. Whole region will be taken into account if not provided. Defaults to None.

返回 The loss value.

返回类型 torch.Tensor

class mmocr.models.common.losses.**MaskedBalancedBCEWithLogitsLoss**(*reduction='none'*, *negative_ratio=3*, *fallback_negative_num=0*, *eps=1e-06*)

This loss combines a Sigmoid layers and a masked balanced BCE loss in one single class. It's AMP-eligible.

参数

- **reduction** (*str, optional*) –The method to reduce the loss. Options are 'none', 'mean' and 'sum'. Defaults to 'none'.
- **negative_ratio** (*float or int, optional*) –Maximum ratio of negative samples to positive ones. Defaults to 3.
- **fallback_negative_num** (*int, optional*) –When the mask contains no positive samples, the number of negative samples to be sampled. Defaults to 0.
- **eps** (*float, optional*) –Eps to avoid zero-division error. Defaults to 1e-6.

返回类型 None

forward(*pred*, *gt*, *mask=None*)

Forward function.

参数

- **pred** (*torch.Tensor*) –The prediction in any shape.
- **gt** (*torch.Tensor*) –The learning target of the prediction in the same shape as pred.
- **mask** (*torch.Tensor, optional*) –Binary mask in the same shape of pred, indicating positive regions to calculate the loss. Whole region will be taken into account if not provided. Defaults to None.

返回 The loss value.

返回类型 torch.Tensor

class mmocr.models.common.losses.**MaskedDiceLoss**(*eps=1e-06*)

Masked dice loss.

参数 **eps** (*float, optional*) –Eps to avoid zero-divison error. Defaults to 1e-6.

返回类型 None

forward(*pred*, *gt*, *mask=None*)

Forward function.

参数

- **pred** (*torch.Tensor*) –The prediction in any shape.

- **gt** (*torch.Tensor*) –The learning target of the prediction in the same shape as pred.
- **mask** (*torch.Tensor, optional*) –Binary mask in the same shape of pred, indicating positive regions to calculate the loss. Whole region will be taken into account if not provided. Defaults to None.

返回 The loss value.

返回类型 torch.Tensor

class mmocr.models.common.losses.**MaskedSmoothL1Loss**(*beta=1*, *eps=1e-06*)

Masked Smooth L1 loss.

参数

- **beta** (*float, optional*) –The threshold in the piecewise function. Defaults to 1.
- **eps** (*float, optional*) –Eps to avoid zero-division error. Defaults to 1e-6.

返回类型 None

forward(*pred*, *gt*, *mask=None*)

Forward function.

参数

- **pred** (*torch.Tensor*) –The prediction in any shape.
- **gt** (*torch.Tensor*) –The learning target of the prediction in the same shape as pred.
- **mask** (*torch.Tensor, optional*) –Binary mask in the same shape of pred, indicating positive regions to calculate the loss. Whole region will be taken into account if not provided. Defaults to None.

返回 The loss value.

返回类型 torch.Tensor

class mmocr.models.common.losses.**MaskedSquareDiceLoss**(*eps=0.001*)

Masked square dice loss.

参数 **eps** (*float, optional*) –Eps to avoid zero-divison error. Defaults to 1e-3.

返回类型 None

forward(*pred*, *gt*, *mask=None*)

Forward function.

参数

- **pred** (*torch.Tensor*) –The prediction in any shape.
- **gt** (*torch.Tensor*) –The learning target of the prediction in the same shape as pred.

- **mask** (*torch.Tensor, optional*) –Binary mask in the same shape of pred, indicating positive regions to calculate the loss. Whole region will be taken into account if not provided. Defaults to None.

返回 The loss value.

返回类型 torch.Tensor

class mmocr.models.common.losses.**SmoothL1Loss**(*size_average=None*, *reduce=None*, *reduction='mean'*, *beta=1.0*)

Smooth L1 loss.

参数

- **reduction** (*str*) –
- **beta** (*float*) –

返回类型 None

class mmocr.models.common.dictionary.**Dictionary**(*dict_file*, *with_start=False*, *with_end=False*, *same_start_end=False*, *with_padding=False*, *with_unknown=False*, *start_token='<BOS>'*, *end_token='<EOS>'*, *start_end_token='<BOS/EOS>'*, *padding_token='<PAD>'*, *unknown_token='<UKN>'*)

The class generates a dictionary for recognition. It pre-defines four special tokens: `start_token`, `end_token`, `pad_token`, and `unknown_token`, which will be sequentially placed at the end of the dictionary when their corresponding flags are True.

参数

- **dict_file** (*str*) –The path of Character dict file which a single character must occupies a line.
- **with_start** (*bool*) –The flag to control whether to include the start token. Defaults to False.
- **with_end** (*bool*) –The flag to control whether to include the end token. Defaults to False.
- **same_start_end** (*bool*) –The flag to control whether the start token and end token are the same. It only works when both `with_start` and `with_end` are True. Defaults to False.
- **with_padding** (*bool*) –The padding token may represent more than a padding. It can also represent tokens like the blank token in CTC or the background token in SegOCR. Defaults to False.
- **with_unknown** (*bool*) –The flag to control whether to include the unknown token. Defaults to False.

- **start_token** (*str*) –The start token as a string. Defaults to '<BOS>'.
- **end_token** (*str*) –The end token as a string. Defaults to '<EOS>'.
- **start_end_token** (*str*) –The start/end token as a string. if start and end is the same. Defaults to '<BOS/EOS>'.
- **padding_token** (*str*) –The padding token as a string. Defaults to '<PAD>'.
- **unknown_token** (*str, optional*) –The unknown token as a string. If it's set to None and `with_unknown` is True, the unknown token will be skipped when converting string to index. Defaults to '<UKN>'.

返回类型 None

char2idx(*char*, *strict=True*)

Convert a character to an index via `Dictionary.dict`.

参数

- **char** (*str*) –The character to convert to index.
- **strict** (*bool*) –The flag to control whether to raise an exception when the character is not in the dictionary. Defaults to True.

返回 The index of the character.

返回类型 int

property dict: list

Returns a list of characters to recognize, where special tokens are counted.

Type list

idx2str(*index*)

Convert a list of index to string.

参数 **index** (*list[int]*) –The list of indexes to convert to string.

返回 The converted string.

返回类型 str

property num_classes: int

Number of output classes. Special tokens are counted.

Type int

str2idx(*string*)

Convert a string to a list of indexes via `Dictionary.dict`.

参数 **string** (*str*) –The string to convert to indexes.

返回 The list of indexes of the string.

返回类型 list

class mmocr.models.common.layers.**TFDecoderLayer**(*d_model=512*, *d_inner=256*, *n_head=8*, *d_k=64*, *d_v=64*, *dropout=0.1*, *qkv_bias=False*, *act_cfg={'type': 'mmengine.GELU'}*, *operation_order=None*)

Transformer Decoder Layer.

参数

- **d_model** (*int*) –The number of expected features in the decoder inputs (default=512).
- **d_inner** (*int*) –The dimension of the feedforward network model (default=256).
- **n_head** (*int*) –The number of heads in the multiheadattention models (default=8).
- **d_k** (*int*) –Total number of features in key.
- **d_v** (*int*) –Total number of features in value.
- **dropout** (*float*) –Dropout layer on attn_output_weights.
- **qkv_bias** (*bool*) –Add bias in projection layer. Default: False.
- **act_cfg** (*dict*) –Activation cfg for feedforward module.
- **operation_order** (*tuple[str]*) –The execution order of operation in transformer. Such as ('self_attn', 'norm', 'enc_dec_attn', 'norm', 'ffn', 'norm') or ('norm', 'self_attn', 'norm', 'enc_dec_attn', 'norm', 'ffn'). Default: None.

forward(*dec_input*, *enc_output*, *self_attn_mask=None*, *dec_enc_attn_mask=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.common.layers.**TFEncoderLayer**(*d_model=512*, *d_inner=256*, *n_head=8*, *d_k=64*, *d_v=64*, *dropout=0.1*, *qkv_bias=False*, *act_cfg={'type': 'mmengine.GELU'}*, *operation_order=None*)

Transformer Encoder Layer.

参数

- **d_model** (*int*) –The number of expected features in the decoder inputs (default=512).
- **d_inner** (*int*) –The dimension of the feedforward network model (default=256).
- **n_head** (*int*) –The number of heads in the multiheadattention models (default=8).

- **d_k** (*int*) –Total number of features in key.
- **d_v** (*int*) –Total number of features in value.
- **dropout** (*float*) –Dropout layer on attn_output_weights.
- **qkv_bias** (*bool*) –Add bias in projection layer. Default: False.
- **act_cfg** (*dict*) –Activation cfg for feedforward module.
- **operation_order** (*tuple[str]*) –The execution order of operation in transformer. Such as (‘self_attn’, ‘norm’, ‘ffn’, ‘norm’) or (‘norm’, ‘self_attn’, ‘norm’, ‘ffn’). Default：None.

forward(*x*, *mask=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解： Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.common.modules.**MultiHeadAttention**(*n_head=8*, *d_model=512*, *d_k=64*, *d_v=64*, *dropout=0.1*, *qkv_bias=False*)

Multi-Head Attention module.

参数

- **n_head** (*int*) –The number of heads in the multiheadattention models (default=8).
- **d_model** (*int*) –The number of expected features in the decoder inputs (default=512).
- **d_k** (*int*) –Total number of features in key.
- **d_v** (*int*) –Total number of features in value.
- **dropout** (*float*) –Dropout layer on attn_output_weights.
- **qkv_bias** (*bool*) –Add bias in projection layer. Default: False.

forward(*q*, *k*, *v*, *mask=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解： Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while

the latter silently ignores them.

---

class mmocr.models.common.modules.**PositionalEncoding**(*d_hid=512*, *n_position=200*, *dropout=0*)

Fixed positional encoding with sine and cosine functions.

forward(*x*)

参数 **x** (*Tensor*) –Tensor of shape (batch_size, pos_len, d_hid, ···)

class mmocr.models.common.modules.**PositionwiseFeedForward**(*d_in*, *d_hid*, *dropout=0.1*, *act_cfg={'type': 'Relu'}*)

Two-layer feed-forward module.

参数

- **d_in** (*int*) –The dimension of the input for feedforward network model.
- **d_hid** (*int*) –The dimension of the feedforward network model.
- **dropout** (*float*) –Dropout layer on feedforward output.
- **act_cfg** (*dict*) –Activation cfg for feedforward module.

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.

---

class mmocr.models.common.modules.**ScaledDotProductAttention**(*temperature*, *attn_dropout=0.1*)

Scaled Dot-Product Attention Module. This code is adopted from https://github.com/jadore801120/attention-is-all-you-need-pytorch.

参数

- **temperature** (*float*) –The scale factor for softmax input.
- **attn_dropout** (*float*) –Dropout layer on attn_output_weights.

forward(*q*, *k*, *v*, *mask=None*)

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.

40.2 Text Detection Detectors

class mmocr.models.textdet.detectors.**DBNet**(*backbone*, *det_head*, *neck=None*, *data_preprocessor=None*, *init_cfg=None*)

The class for implementing DBNet text detector: Real-time Scene Text Detection with Differentiable Binarization.

[https://arxiv.org/abs/1911.08947].

参数

- **backbone** (*Dict*) –
- **det_head** (*Dict*) –
- **neck** (*Optional[Dict]*) –
- **data_preprocessor** (*Optional[Dict]*) –
- **init_cfg** (*Optional[Dict]*) –

返回类型 None

class mmocr.models.textdet.detectors.**DRRG**(*backbone*, *det_head*, *neck=None*, *data_preprocessor=None*, *init_cfg=None*)

The class for implementing DRRG text detector. Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection.

[https://arxiv.org/abs/2003.07493]

参数

- **backbone** (*Dict*) –
- **det_head** (*Dict*) –
- **neck** (*Optional[Dict]*) –
- **data_preprocessor** (*Optional[Dict]*) –
- **init_cfg** (*Optional[Dict]*) –

返回类型 None

class mmocr.models.textdet.detectors.**FCENet**(*backbone*, *det_head*, *neck=None*, *data_preprocessor=None*, *init_cfg=None*)

The class for implementing FCENet text detector FCENet(CVPR2021): Fourier Contour Embedding for Arbitrary-shaped Text

Detection

[https://arxiv.org/abs/2104.10442]

参数

- **backbone** (*Dict*) –
- **det_head** (*Dict*) –
- **neck** (*Optional[Dict]*) –
- **data_preprocessor** (*Optional[Dict]*) –
- **init_cfg** (*Optional[Dict]*) –

返回类型 None

class mmocr.models.textdet.detectors.**MMDetWrapper**(*cfg*, *text_repr_type='poly'*)

A wrapper of MMDet's model.

参数

- **cfg** (*dict*) –The config of the model.
- **text_repr_type** (*str*) –The boundary encoding type 'poly' or 'quad'. Defaults to 'poly'.

返回类型 None

adapt_predictions(*data*, *data_samples*)

Convert Instance datas from MMDet into MMOCR's format.

参数

- **data** (*List[mmdet.structures.det_data_sample.DetDataSample]*) – (list[DetDataSample]): Detection results of the input images. Each DetDataSample usually contain 'pred_instances'. And the `pred_instances` usually contains following keys. - scores (Tensor): Classification scores, has a shape

  (num_instance,)

  - **labels (Tensor): Labels of bboxes, has a shape** (num_instances,).
  - **bboxes (Tensor): Has a shape (num_instances, 4),** the last dimension 4 arrange as (x1, y1, x2, y2).
  - masks (Tensor, Optional): Has a shape (num_instances, H, W).

- **data_samples** (list[TextDetDataSample]) –The annotation data of every samples.

返回

A list of N datasamples containing ground truth and prediction results. The polygon results are saved in TextDetDataSample.pred_instances.polygons The confidence scores are saved in TextDetDataSample.pred_instances.scores.

返回类型 list[*TextDetDataSample*]

forward(*inputs*, *data_samples=None*, *mode='tensor'*, ***kwargs*)

The unified entry for a forward process in both training and test.

The method works in three modes: “tensor”, “predict” and “loss”:

- “tensor”: Forward the whole network and return tensor or tuple of

tensor without any post-processing, same as a common nn.Module. - “predict”: Forward and return the predictions, which are fully processed to a list of DetDataSample. - “loss”: Forward and return a dict of losses according to the given inputs and data samples.

Note that this method doesn’t handle either back propagation or parameter update, which are supposed to be done in train_step().

参数

- **inputs** (*torch.Tensor*) –The input tensor with shape (N, C, …) in general.
- **data_samples** (*Optional[Union[List[mmocr.structures.textdet_data_sample.TextDetDataSample], List[mmdet.structures.det_data_sample.DetDataSample]]]*) –
- **mode** (*str*) –

返回类型 Union[Dict[str, torch.Tensor], List[mmdet.structures.det_data_sample.DetDataSample], Tuple[torch.Tensor], torch.Tensor]

:param data_samples (list[DetDataSample] or: list[TextDetDataSample]): The annotation data of every sample. When in “predict” mode, it should be a list of TextDetDataSample. Otherwise they are :obj:‘DetDataSample‘s. Defaults to None.

参数

- **mode** (*str*) –Running mode. Defaults to ‘tensor’.
- **inputs** (*torch.Tensor*) –
- **data_samples** (*Optional[Union[List[mmocr.structures.textdet_data_sample.TextDetDataSample], List[mmdet.structures.det_data_sample.DetDataSample]]]*) –

返回

The return type depends on `mode`.

- If `mode="tensor"`, return a tensor or a tuple of tensor.
- If `mode="predict"`, return a list of `TextDetDataSample`.
- If `mode="loss"`, return a dict of tensor.

返回类型 Union[Dict[str, torch.Tensor], List[mmdet.structures.det_data_sample.DetDataSample], Tuple[torch.Tensor], torch.Tensor]

class mmocr.models.textdet.detectors.**PANet**(*backbone*, *det_head*, *neck=None*, *data_preprocessor=None*, *init_cfg=None*)

The class for implementing PANet text detector:

Efficient and Accurate Arbitrary-Shaped Text Detection with Pixel Aggregation Network [https://arxiv.org/abs/1908.05900].

参数

- **backbone** (*Dict*) –
- **det_head** (*Dict*) –
- **neck** (*Optional[Dict]*) –
- **data_preprocessor** (*Optional[Dict]*) –
- **init_cfg** (*Optional[Dict]*) –

返回类型 None

class mmocr.models.textdet.detectors.**PSENet**(*backbone*, *det_head*, *neck=None*, *data_preprocessor=None*, *init_cfg=None*)

The class for implementing PSENet text detector: Shape Robust Text Detection with Progressive Scale Expansion Network.

[https://arxiv.org/abs/1806.02559].

参数

- **backbone** (*Dict*) –
- **det_head** (*Dict*) –
- **neck** (*Optional[Dict]*) –
- **data_preprocessor** (*Optional[Dict]*) –
- **init_cfg** (*Optional[Dict]*) –

返回类型 None

class mmocr.models.textdet.detectors.**SingleStageTextDetector**(*backbone*, *det_head*, *neck=None*, *data_preprocessor=None*, *init_cfg=None*)

The class for implementing single stage text detector.

Single-stage text detectors directly and densely predict bounding boxes or polygons on the output features of the backbone + neck (optional).

参数

- **backbone** (*dict*) –Backbone config.
- **neck** (*dict, optional*) –Neck config. If None, the output from backbone will be directly fed into det_head.
- **det_head** (*dict*) –Head config.
- **data_preprocessor** (*dict, optional*) –Model preprocessing config for processing the input image data. Keys allowed are "to_rgb"(bool), "pad_size_divisor"(int), "pad_value"(int or float), "mean"(int or float) and "std"(int or float). Preprcessing order: 1. to rgb; 2. normalization 3. pad. Defaults to None.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

extract_feat(*inputs*)

Extract features.

参数 **inputs** (*Tensor*) –Image tensor with shape (N, C, H ,W).

返回 Multi-level features that may have different resolutions.

返回类型 Tensor or tuple[Tensor]

loss(*inputs*, *data_samples*)

Calculate losses from a batch of inputs and data samples.

参数

- **inputs** (*torch.Tensor*) –Input images of shape (N, C, H, W). Typically these should be mean centered and std scaled.
- **data_samples** (*list[TextDetDataSample]*) –A list of N datasamples, containing meta information and gold annotations for each of the images.

返回 A dictionary of loss components.

返回类型 dict[str, Tensor]

predict(*inputs*, *data_samples*)

Predict results from a batch of inputs and data samples with post- processing.

参数

- **inputs** (*torch.Tensor*) –Images of shape (N, C, H, W).
- **data_samples** (*list[TextDetDataSample]*) –A list of N datasamples, containing meta information and gold annotations for each of the images.

返回

A list of N datasamples of prediction results. Each DetDataSample usually contain 'pred_instances'. And the `pred_instances` usually contains following keys.

- **scores (Tensor): Classification scores, has a shape** (num_instance,)
- **labels (Tensor): Labels of bboxes, has a shape** (num_instances,).
- **bboxes (Tensor): Has a shape (num_instances, 4),** the last dimension 4 arrange as (x1, y1, x2, y2).
- **polygons (list[np.ndarray]): The length is num_instances.** Each element represents the polygon of the instance, in (xn, yn) order.

返回类型 list[*TextDetDataSample*]

class mmocr.models.textdet.detectors.**TextSnake**(*backbone*, *det_head*, *neck=None*, *data_preprocessor=None*, *init_cfg=None*)

The class for implementing TextSnake text detector: TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

[https://arxiv.org/abs/1807.01544]

参数

- **backbone** (*Dict*) –
- **det_head** (*Dict*) –
- **neck** (*Optional[Dict]*) –
- **data_preprocessor** (*Optional[Dict]*) –
- **init_cfg** (*Optional[Dict]*) –

返回类型 None

40.3 Text Detection Heads

class mmocr.models.textdet.heads.**BaseTextDetHead**(*module_loss*, *postprocessor*, *init_cfg=None*)

Base head for text detection, build the loss and postprocessor.

1. The `init_weights` method is used to initialize head's model parameters. After detector initialization, `init_weights` is triggered when `detector.init_weights()` is called externally.

2. The `loss` method is used to calculate the loss of head, which includes two steps: (1) the head model performs forward propagation to obtain the feature maps (2) The `module_loss` method is called based on the feature maps to calculate the loss.

```
loss(): forward() -> module_loss()
```

3. The `predict` method is used to predict detection results, which includes two steps: (1) the head model performs forward propagation to obtain the feature maps (2) The `postprocessor` method is called based on the feature maps to predict detection results including post-processing.

```
predict(): forward() -> postprocessor()
```

4. The `loss_and_predict` method is used to return loss and detection results at the same time. It will call head's `forward`, `module_loss` and `postprocessor` methods in order.

```
loss_and_predict(): forward() -> module_loss() -> postprocessor()
```

参数

- **loss** (*dict*) –Config to build loss.
- **postprocessor** (*dict*) –Config to build postprocessor.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.
- **module_loss** (*Dict*) –

返回类型 None

loss(*x*, *data_samples*)

Perform forward propagation and loss calculation of the detection head on the features of the upstream network.

参数

- **x** (*tuple[Tensor]*) –Features from the upstream network, each is a 4D-tensor.
- **data_samples** (List[DetDataSample]) –The Data Samples. It usually includes information such as *gt_instance*, *gt_panoptic_seg* and *gt_sem_seg*.

返回 A dictionary of loss components.

返回类型 dict

loss_and_predict(*x*, *data_samples*)

Perform forward propagation of the head, then calculate loss and predictions from the features and data samples.

参数

- **x** (*tuple[Tensor]*) –Features from FPN.
- **data_samples** (list[DetDataSample]) –Each item contains the meta information of each image and corresponding annotations.

返回

the return value is a tuple contains:

- losses: (dict[str, Tensor]): A dictionary of loss components.
- predictions (list[InstanceData]): Detection results of each image after the post process.

返回类型 tuple

predict(*x*, *data_samples*)

Perform forward propagation of the detection head and predict detection results on the features of the upstream network.

参数

- **x** (*tuple[Tensor]*) –Multi-level features from the upstream network, each is a 4D-tensor.
- **data_samples** (List[DetDataSample]) –The Data Samples. It usually includes information such as *gt_instance*, *gt_panoptic_seg* and *gt_sem_seg*.

返回 Detection results of each image after the post process.

返回类型 SampleList

class mmocr.models.textdet.heads.**DBHead**(*in_channels*, *with_bias=False*, *module_loss={'type': 'DBModuleLoss'}*, *postprocessor={'text_repr_type': 'quad', 'type': 'DBPostprocessor'}*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv'}, {'type': 'Constant', 'layer': 'BatchNorm', 'val': 1.0, 'bias': 0.0001}]*)

The class for DBNet head.

This was partially adapted from https://github.com/MhLiao/DB

参数

- **in_channels** (*int*) –The number of input channels.

- **with_bias** (*bool*) –Whether add bias in Conv2d layer. Defaults to False.
- **module_loss** (*dict*) –Config of loss for dbnet. Defaults to `dict(type='DBModuleLoss')`
- **postprocessor** (*dict*) –Config of postprocessor for dbnet.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

forward(*img*, *data_samples=None*, *mode='predict'*)

参数

- **img** (*Tensor*) –Shape (N, C, H, W).
- **data_samples** (*list[TextDetDataSample], optional*) –A list of data samples. Defaults to None.
- **mode** (*str*) –Forward mode. It affects the return values. Options are "loss", "predict" and "both". Defaults to "predict".
  - `loss`: Run the full network and return the prob logits, threshold map and binary map.
  - `predict`: Run the binarzation part and return the prob map only.
  - `both`: Run the full network and return prob logits, threshold map, binary map and prob map.

返回 Its type depends on `mode`, read its docstring for details. Each has the shape of $(N, 4H, 4W)$.

返回类型 Tensor or tuple(Tensor)

loss(*x*, *batch_data_samples*)

Perform forward propagation and loss calculation of the detection head on the features of the upstream network.

参数

- **x** (*tuple[Tensor]*) –Features from the upstream network, each is a 4D-tensor.
- **batch_data_samples** (List[`DetDataSample`]) –The Data Samples. It usually includes information such as *gt_instance*, *gt_panoptic_seg* and *gt_sem_seg*.

返回 A dictionary of loss components.

返回类型 dict

loss_and_predict(*x*, *batch_data_samples*)

Perform forward propagation of the head, then calculate loss and predictions from the features and data samples.

参数

- **x** (*tuple[Tensor]*) –Features from FPN.
- **batch_data_samples** (list[DetDataSample]) –Each item contains the meta information of each image and corresponding annotations.

返回

the return value is a tuple contains:

- losses: (dict[str, Tensor]): A dictionary of loss components.
- predictions (list[InstanceData]): Detection results of each image after the post process.

返回类型 tuple

predict(*x*, *batch_data_samples*)

Perform forward propagation of the detection head and predict detection results on the features of the upstream network.

参数

- **x** (*tuple[Tensor]*) –Multi-level features from the upstream network, each is a 4D-tensor.
- **batch_data_samples** (List[DetDataSample]) –The Data Samples. It usually includes information such as *gt_instance*, *gt_panoptic_seg* and *gt_sem_seg*.

返回 Detection results of each image after the post process.

返回类型 SampleList

class mmocr.models.textdet.heads.**DRRGHead**(*in_channels*, *k_at_hops=(8, 4)*, *num_adjacent_linkages=3*, *node_geo_feat_len=120*, *pooling_scale=1.0*, *pooling_output_size=(4, 3)*, *nms_thr=0.3*, *min_width=8.0*, *max_width=24.0*, *comp_shrink_ratio=1.03*, *comp_ratio=0.4*, *comp_score_thr=0.3*, *text_region_thr=0.2*, *center_region_thr=0.2*, *center_region_area_thr=50*, *local_graph_thr=0.7*, *module_loss={'type': 'DRRGModuleLoss'}*, *postprocessor={'link_thr': 0.85, 'type': 'DRRGPostprocessor'}*, *init_cfg={'mean': 0, 'override': {'name': 'out_conv'}, 'std': 0.01, 'type': 'Normal'}*)

The class for DRRG head: Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection.

参数

- **in_channels** (*int*) –The number of input channels.
- **k_at_hops** (*tuple(int)*) –The number of i-hop neighbors, i = 1, 2. Defaults to (8, 4).
- **num_adjacent_linkages** (*int*) –The number of linkages when constructing adjacent matrix. Defaults to 3.
- **node_geo_feat_len** (*int*) –The length of embedded geometric feature vector of a component. Defaults to 120.
- **pooling_scale** (*float*) –The spatial scale of rotated RoI-Align. Defaults to 1.0.
- **pooling_output_size** (*tuple(int)*) –The output size of RRoI-Aligning. Defaults to (4, 3).
- **nms_thr** (*float*) –The locality-aware NMS threshold of text components. Defaults to 0.3.
- **min_width** (*float*) –The minimum width of text components. Defaults to 8.0.
- **max_width** (*float*) –The maximum width of text components. Defaults to 24.0.
- **comp_shrink_ratio** (*float*) –The shrink ratio of text components. Defaults to 1.03.
- **comp_ratio** (*float*) –The reciprocal of aspect ratio of text components. Defaults to 0.4.
- **comp_score_thr** (*float*) –The score threshold of text components. Defaults to 0.3.
- **text_region_thr** (*float*) –The threshold for text region probability map. Defaults to 0.2.
- **center_region_thr** (*float*) –The threshold for text center region probability map. Defaults to 0.2.
- **center_region_area_thr** (*int*) –The threshold for filtering small-sized text center region. Defaults to 50.
- **local_graph_thr** (*float*) –The threshold to filter identical local graphs. Defaults to 0.7.
- **module_loss** (*dict*) –The config of loss that DRRGHead uses. Defaults to `dict(type='DRRGModuleLoss')`.
- **postprocessor** (*dict*) –Config of postprocessor for Drrg. Defaults to `dict(type='DrrgPostProcessor', link_thr=0.85)`.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to `dict(type='Normal', override=dict(name='out_conv'), mean=0, std=0.01)`.

返回类型 None

forward(*inputs*, *data_samples=None*)

Run DRRG head in prediction mode, and return the raw tensors only. :param inputs: Shape of $(1, C, H, W)$. :type inputs: Tensor :param data_samples: A list of data

samples. Defaults to None.

返回

Returns (edge, score, text_comps).

- edge (ndarray): The edge array of shape $(N_{edges}, 2)$ where each row is a pair of text component indices that makes up an edge in graph.
- score (ndarray): The score array of shape $(N_{edges},)$, corresponding to the edge above.
- text_comps (ndarray): The text components of shape $(M, 9)$ where each row corresponds to one box and its score: (x1, y1, x2, y2, x3, y3, x4, y4, score).

返回类型 tuple

参数

- **inputs** (*torch.Tensor*) –
- **data_samples** (*list[TextDetDataSample], optional*) –

loss(*inputs*, *data_samples*)

Loss function.

参数

- **inputs** (*Tensor*) –Shape of (N, C, H, W).
- **data_samples** (*List[TextDetDataSample]*) –List of data samples.

返回

- **pred_maps (Tensor): Prediction map with shape** $(N, 6, H, W)$.
- **gcn_pred (Tensor): Prediction from GCN module, with** shape $(N, 2)$.
- **gt_labels (Tensor): Ground-truth label of shape** (m, n) where $m * n = N$.

返回类型 tuple(pred_maps, gcn_pred, gt_labels)

class mmocr.models.textdet.heads.**FCEHead**(*in_channels*, *fourier_degree=5*, *module_loss={'num_sample': 50, 'type': 'FCEModuleLoss'}*, *postprocessor={'alpha': 1.0, 'beta': 2.0, 'num_reconstr_points': 50, 'score_thr': 0.3, 'text_repr_type': 'poly', 'type': 'FCEPostprocessor'}*, *init_cfg={'mean': 0, 'override': [{'name': 'out_conv_cls'}, {'name': 'out_conv_reg'}], 'std': 0.01, 'type': 'Normal'}*)

The class for implementing FCENet head.

FCENet(CVPR2021): Fourier Contour Embedding for Arbitrary-shaped Text Detection

参数

- **in_channels** (*int*) –The number of input channels.
- **fourier_degree** (*int*) –The maximum Fourier transform degree k. Defaults to 5.
- **module_loss** (*dict*) –Config of loss for FCENet. Defaults to `dict(type='FCEModuleLoss', num_sample=50)`.
- **postprocessor** (*dict*) –Config of postprocessor for FCENet.
- **init_cfg** (*dict, optional*) –Initialization configs.

返回类型 None

forward(*inputs*, *data_samples=None*)

参数

- **inputs** (*List[Tensor]*) –Each tensor has the shape of (N, C_i, H_i, W_i).
- **data_samples** (*list[TextDetDataSample], optional*) –A list of data samples. Defaults to None.

返回 A list of dict with keys of `cls_res`, `reg_res` corresponds to the classification result and regression result computed from the input tensor with the same index. They have the shapes of $(N, C_{cls,i}, H_i, W_i)$ and $(N, C_{out,i}, H_i, W_i)$.

返回类型 list[dict]

forward_single(*x*)

Forward function for a single feature level.

参数 **x** (*Tensor*) –The input tensor with the shape of (N, C_i, H_i, W_i).

返回 The classification and regression result with the shape of $(N, C_{cls,i}, H_i, W_i)$ and $(N, C_{out,i}, H_i, W_i)$.

返回类型 Tensor

class mmocr.models.textdet.heads.**PANHead**(*in_channels*, *hidden_dim*, *out_channel*, *module_loss={'type': 'PANModuleLoss'}*, *postprocessor={'text_repr_type': 'poly', 'type': 'PANPostprocessor'}*, *init_cfg=[{'type': 'Normal', 'mean': 0, 'std': 0.01, 'layer': 'Conv2d'}, {'type': 'Constant', 'val': 1, 'bias': 0, 'layer': 'BN'}]*)

The class for PANet head.

参数

- **in_channels** (*list[int]*) –A list of 4 numbers of input channels.
- **hidden_dim** (*int*) –The hidden dimension of the first convolutional layer.
- **out_channel** (*int*) –Number of output channels.
- **module_loss** (*dict*) –Configuration dictionary for loss type. Defaults to dict(type='PANModuleLoss')
- **postprocessor** (*dict*) –Config of postprocessor for PANet. Defaults to dict(type='PANPostprocessor', text_repr_type='poly').
- **init_cfg** (*list[dict]*) –Initialization configs. Defaults to [dict(type='Normal', mean=0, std=0.01, layer='Conv2d'),

  dict(type='Constant', val=1, bias=0, layer='BN')]

返回类型 None

forward(*inputs*, *data_samples=None*)

PAN head forward. :param inputs: Each tensor has the shape of

(N, C_i, W, H), where $\sum_i C_i = C_{in}$ and C_{in} is `input_channels`.

参数

- **data_samples** (*list[TextDetDataSample], optional*) –A list of data samples. Defaults to None.
- **inputs** (*list[Tensor] | Tensor*) –

返回 A tensor of shape (N, C_{out}, W, H) where C_{out} is `output_channels`.

返回类型 Tensor

class mmocr.models.textdet.heads.**PSEHead**(*in_channels*, *hidden_dim*, *out_channel*, *module_loss={'type': 'PSEModuleLoss'}*, *postprocessor={'text_repr_type': 'poly', 'type': 'PSEPostprocessor'}*, *init_cfg=None*)

The class for PSENet head.

参数

- **in_channels** (*list[int]*) –A list of numbers of input channels.
- **hidden_dim** (*int*) –The hidden dimension of the first convolutional layer.
- **out_channel** (*int*) –Number of output channels.
- **module_loss** (*dict*) –Configuration dictionary for loss type. Supported loss types are "PANModuleLoss" and "PSEModuleLoss". Defaults to PSEModuleLoss.
- **postprocessor** (*dict*) –Config of postprocessor for PSENet.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

class mmocr.models.textdet.heads.**TextSnakeHead**(*in_channels*, *out_channels=5*, *downsample_ratio=1.0*, *module_loss={'type': 'TextSnakeModuleLoss'}*, *postprocessor={'text_repr_type': 'poly', 'type': 'TextSnakePostprocessor'}*, *init_cfg={'mean': 0, 'override': {'name': 'out_conv'}, 'std': 0.01, 'type': 'Normal'}*)

The class for TextSnake head: TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

参数

- **in_channels** (*int*) –Number of input channels.
- **out_channels** (*int*) –Number of output channels.
- **downsample_ratio** (*float*) –Downsample ratio.
- **module_loss** (*dict*) –Configuration dictionary for loss type. Defaults to `dict(type='TextSnakeModuleLoss')`.
- **postprocessor** (*dict*) –Config of postprocessor for TextSnake.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

forward(*inputs*, *data_samples=None*)

参数

- **inputs** (*torch.Tensor*) –Shape (N, C_{in}, H, W), where C_{in} is `in_channels`. H and W should be the same as the input of backbone.
- **data_samples** (*list[TextDetDataSample], optional*) –A list of data samples. Defaults to None.

返回 A tensor of shape $(N, 5, H, W)$, where the five channels represent [0]: text score, [1]: center score, [2]: sin, [3] cos, [4] radius, respectively.

返回类型 Tensor

40.4 Text Detection Necks

class mmocr.models.textdet.necks.**FPEM_FFM**(*in_channels*, *conv_out=128*, *fpem_repeat=2*, *align_corners=False*, *init_cfg={'distribution': 'uniform', 'layer': 'Conv2d', 'type': 'Xavier'}*)

This code is from https://github.com/WenmuZhou/PAN.pytorch.

参数

- **in_channels** (*list[int]*) –A list of 4 numbers of input channels.
- **conv_out** (*int*) –Number of output channels.
- **fpem_repeat** (*int*) –Number of FPEM layers before FFM operations.
- **align_corners** (*bool*) –The interpolation behaviour in FFM operation, used in `torch.nn.functional.interpolate()`.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*x*)

参数 **x** (*list[Tensor]*) –A list of four tensors of shape (N, C_i, H_i, W_i), representing C2, C3, C4, C5 features respectively. C_i should matches the number in `in_channels`.

返回 Four tensors of shape (N, C_{out}, H_0, W_0) where C_{out} is `conv_out`.

返回类型 list[Tensor]

class mmocr.models.textdet.necks.**FPNC**(*in_channels*, *lateral_channels=256*, *out_channels=64*, *bias_on_lateral=False*, *bn_re_on_lateral=False*, *bias_on_smooth=False*, *bn_re_on_smooth=False*, *asf_cfg=None*, *conv_after_concat=False*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv'}, {'type': 'Constant', 'layer': 'BatchNorm', 'val': 1.0, 'bias': 0.0001}]*)

FPN-like fusion module in Real-time Scene Text Detection with Differentiable Binarization.

This was partially adapted from https://github.com/MhLiao/DB and https://github.com/WenmuZhou/DBNet.pytorch.

参数

- **in_channels** (*list[int]*) –A list of numbers of input channels.

- **lateral_channels** (*int*) –Number of channels for lateral layers.
- **out_channels** (*int*) –Number of output channels.
- **bias_on_lateral** (*bool*) –Whether to use bias on lateral convolutional layers.
- **bn_re_on_lateral** (*bool*) –Whether to use BatchNorm and ReLU on lateral convolutional layers.
- **bias_on_smooth** (*bool*) –Whether to use bias on smoothing layer.
- **bn_re_on_smooth** (*bool*) –Whether to use BatchNorm and ReLU on smoothing layer.
- **asf_cfg** (*dict, optional*) –Adaptive Scale Fusion module configs. The attention_type can be 'ScaleChannelSpatial'.
- **conv_after_concat** (*bool*) –Whether to add a convolution layer after the concatenation of predictions.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

forward(*inputs*)

参数 **inputs** (*list[Tensor]*) –Each tensor has the shape of (N, C_i, H_i, W_i). It usually expects 4 tensors (C2-C5 features) from ResNet.

返回 A tensor of shape (N, C_{out}, H_0, W_0) where C_{out} is `out_channels`.

返回类型 Tensor

class mmocr.models.textdet.necks.**FPNF**(*in_channels=[256, 512, 1024, 2048]*, *out_channels=256*, *fusion_type='concat'*, *init_cfg={'distribution': 'uniform', 'layer': 'Conv2d', 'type': 'Xavier'}*)

FPN-like fusion module in Shape Robust Text Detection with Progressive Scale Expansion Network.

参数

- **in_channels** (*list[int]*) –A list of number of input channels. Defaults to [256, 512, 1024, 2048].
- **out_channels** (*int*) –The number of output channels. Defaults to 256.
- **fusion_type** (*str*) –Type of the final feature fusion layer. Available options are "concat" and "add". Defaults to "concat".
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to dict(type='Xavier', layer='Conv2d', distribution='uniform')

返回类型 None

forward(*inputs*)

参数 **inputs** (*list[Tensor]*) –Each tensor has the shape of (N, C_i, H_i, W_i). It usually expects 4 tensors (C2-C5 features) from ResNet.

返回 A tensor of shape (N, C_{out}, H_0, W_0) where C_{out} is `out_channels`.

返回类型 Tensor

class mmocr.models.textdet.necks.**FPN_UNet**(*in_channels*, *out_channels*, *init_cfg={'distribution': 'uniform', 'layer': ['Conv2d', 'ConvTranspose2d'], 'type': 'Xavier'}*)

The class for implementing DRRG and TextSnake U-Net-like FPN.

DRRG: Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection.

TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

参数

- **in_channels** (*list[int]*) –Number of input channels at each scale. The length of the list should be 4.
- **out_channels** (*int*) –The number of output channels.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

forward(*x*)

参数 **x** (*list[Tensor] | tuple[Tensor]*) –A list of four tensors of shape (N, C_i, H_i, W_i), representing C2, C3, C4, C5 features respectively. C_i should matches the number in `in_channels`.

返回 Shape (N, C, H, W) where $H = 4H_0$ and $W = 4W_0$.

返回类型 Tensor

40.5 Text Detection Module Losses

class mmocr.models.textdet.module_losses.**DBModuleLoss**(*loss_prob={'type': 'MaskedBalancedBCEWithLogitsLoss'}*, *loss_thr={'beta': 0, 'type': 'MaskedSmoothL1Loss'}*, *loss_db={'type': 'MaskedDiceLoss'}*, *weight_prob=5.0*, *weight_thr=10.0*, *shrink_ratio=0.4*, *thr_min=0.3*, *thr_max=0.7*, *min_sidelength=8*)

The class for implementing DBNet loss.

This is partially adapted from https://github.com/MhLiao/DB.

参数

- **loss_prob** (*dict*) –The loss config for probability map. Defaults to dict(type='MaskedBalancedBCEWithLogitsLoss').
- **loss_thr** (*dict*) –The loss config for threshold map. Defaults to dict(type='MaskedSmoothL1Loss', beta=0).
- **loss_db** (*dict*) –The loss config for binary map. Defaults to dict(type='MaskedDiceLoss').
- **weight_prob** (*float*) –The weight of probability map loss. Denoted as α in paper. Defaults to 5.
- **weight_thr** (*float*) –The weight of threshold map loss. Denoted as β in paper. Defaults to 10.
- **shrink_ratio** (*float*) –The ratio of shrunk text region. Defaults to 0.4.
- **thr_min** (*float*) –The minimum threshold map value. Defaults to 0.3.
- **thr_max** (*float*) –The maximum threshold map value. Defaults to 0.7.
- **min_sidelength** (*int or float*) –The minimum sidelength of the minimum rotated rectangle around any text region. Defaults to 8.

返回类型 None

forward(*preds*, *data_samples*)

Compute DBNet loss.

参数

- **preds** (*tuple(tensor)*) –Raw predictions from model, containing `prob_logits`, `thr_map` and `binary_map`. Each is a tensor of shape (N, H, W).
- **data_samples** (*list[TextDetDataSample]*) –The data samples.

返回 The dict for dbnet losses with loss_prob, loss_db and loss_thr.

返回类型 results(dict)

get_targets(*data_samples*)

Generate loss targets from data samples.

参数 **data_samples** (*list* (TextDetDataSample)) –Ground truth data samples.

返回 A tuple of four tensors as DBNet targets.

返回类型 tuple

class mmocr.models.textdet.module_losses.**DRRGModuleLoss**(*ohem_ratio=3.0*, *downsample_ratio=1.0*, *orientation_thr=2.0*, *resample_step=8.0*, *num_min_comps=9*, *num_max_comps=600*, *min_width=8.0*, *max_width=24.0*, *center_region_shrink_ratio=0.3*, *comp_shrink_ratio=1.0*, *comp_w_h_ratio=0.3*, *text_comp_nms_thr=0.25*, *min_rand_half_height=8.0*, *max_rand_half_height=24.0*, *jitter_level=0.2*, *loss_text={'eps': 1e-05, 'fallback_negative_num': 100, 'type': 'MaskedBalanced-BCEWithLogitsLoss'}*, *loss_center={'type': 'MaskedBCEWithLogitsLoss'}*, *loss_top={'reduction': 'none', 'type': 'SmoothL1Loss'}*, *loss_btm={'reduction': 'none', 'type': 'SmoothL1Loss'}*, *loss_sin={'type': 'MaskedSmoothL1Loss'}*, *loss_cos={'type': 'MaskedSmoothL1Loss'}*, *loss_gcn={'type': 'CrossEntropyLoss'}*)

The class for implementing DRRG loss. This is partially adapted from https://github.com/GXYM/DRRG licensed under the MIT license.

DRRG: Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection.

参数

- **ohem_ratio** (*float*) –The negative/positive ratio in ohem. Defaults to 3.0.
- **downsample_ratio** (*float*) –Downsample ratio. Defaults to 1.0. TODO: remove it.
- **orientation_thr** (*float*) –The threshold for distinguishing between head edge and tail edge among the horizontal and vertical edges of a quadrangle. Defaults to 2.0.
- **resample_step** (*float*) –The step size for resampling the text center line. Defaults to 8.0.
- **num_min_comps** (*int*) –The minimum number of text components, which should be larger than k_hop1 mentioned in paper. Defaults to 9.
- **num_max_comps** (*int*) –The maximum number of text components. Defaults to 600.
- **min_width** (*float*) –The minimum width of text components. Defaults to 8.0.
- **max_width** (*float*) –The maximum width of text components. Defaults to 24.0.
- **center_region_shrink_ratio** (*float*) –The shrink ratio of text center regions. Defaults to 0.3.
- **comp_shrink_ratio** (*float*) –The shrink ratio of text components. Defaults to 1.0.
- **comp_w_h_ratio** (*float*) –The width to height ratio of text components. Defaults to 0.3.
- **min_rand_half_height** (*float*) –The minimum half-height of random text components. Defaults to 8.0.
- **max_rand_half_height** (*float*) –The maximum half-height of random text components. Defaults to 24.0.
- **jitter_level** (*float*) –The jitter level of text component geometric features. Defaults to 0.2.
- **loss_text** (*dict*) –The loss config used to calculate the text loss. Defaults to `dict(type='MaskedBalancedBCEWithLogitsLoss', fallback_negative_num=100, eps=1e-5)`.
- **loss_center** (*dict*) –The loss config used to calculate the center loss. Defaults to `dict(type='MaskedBCEWithLogitsLoss')`.
- **loss_top** (*dict*) –The loss config used to calculate the top loss, which is a part of the height loss. Defaults to `dict(type='SmoothL1Loss', reduction='none')`.
- **loss_btm** (*dict*) –The loss config used to calculate the bottom loss, which is a part of the height loss. Defaults to `dict(type='SmoothL1Loss', reduction='none')`.

- **loss_sin** (*dict*) –The loss config used to calculate the sin loss. Defaults to `dict(type='MaskedSmoothL1Loss')`.
- **loss_cos** (*dict*) –The loss config used to calculate the cos loss. Defaults to `dict(type='MaskedSmoothL1Loss')`.
- **loss_gcn** (*dict*) –The loss config used to calculate the GCN loss. Defaults to `dict(type='CrossEntropyLoss')`.
- **text_comp_nms_thr** (*float*) –

返回类型 None

forward(*preds*, *data_samples*)

Compute Drrg loss.

参数

- **preds** (*tuple*) –The prediction tuple(pred_maps, gcn_pred, gt_labels), each of shape $(N, 6, H, W)$, $(N, 2)$ and (m, n), where $m * n = N$.
- **data_samples** (*list[TextDetDataSample]*) –The data samples.

返回 A loss dict with `loss_text`, `loss_center`, `loss_height`, `loss_sin`, `loss_cos`, and `loss_gcn`.

返回类型 dict

get_targets(*data_samples*)

Generate loss targets from data samples.

参数 **data_samples** (*list(TextDetDataSample)*) –Ground truth data samples.

返回 A tuple of 8 lists of tensors as DRRG targets. Read docstring of `_get_target_single` for more details.

返回类型 tuple

class mmocr.models.textdet.module_losses.**FCEModuleLoss**(*fourier_degree*, *num_sample*, *negative_ratio=3.0*, *resample_step=4.0*, *center_region_shrink_ratio=0.3*, *level_size_divisors=(8, 16, 32)*, *level_proportion_range=((0, 0.4), (0.3, 0.7), (0.6, 1.0))*, *loss_tr={'type': 'MaskedBalancedBCELoss'}*, *loss_tcl={'type': 'MaskedBCELoss'}*, *loss_reg_x={'reduction': 'none', 'type': 'SmoothL1Loss'}*, *loss_reg_y={'reduction': 'none', 'type': 'SmoothL1Loss'}*)

The class for implementing FCENet loss.

FCENet(CVPR2021): Fourier Contour Embedding for Arbitrary-shaped Text Detection

参数

- **fourier_degree** (*int*) –The maximum Fourier transform degree k.
- **num_sample** (*int*) –The sampling points number of regression loss. If it is too small, fcenet tends to be overfitting.
- **negative_ratio** (*float or int*) –Maximum ratio of negative samples to positive ones in OHEM. Defaults to 3.
- **resample_step** (*float*) –The step size for resampling the text center line (TCL). It's better not to exceed half of the minimum width.
- **center_region_shrink_ratio** (*float*) –The shrink ratio of text center region.
- **level_size_divisors** (*tuple(int)*) –The downsample ratio on each level.
- **level_proportion_range** (*tuple(tuple(int))*) –The range of text sizes assigned to each level.
- **loss_tr** (*dict*) –The loss config used to calculate the text region loss. Defaults to dict(type='MaskedBalancedBCELoss').
- **loss_tcl** (*dict*) –The loss config used to calculate the text center line loss. Defaults to dict(type='MaskedBCELoss').
- **loss_reg_x** (*dict*) –The loss config used to calculate the regression loss on x axis. Defaults to dict(type='MaskedSmoothL1Loss').
- **loss_reg_y** (*dict*) –The loss config used to calculate the regression loss on y axis. Defaults to dict(type='MaskedSmoothL1Loss').

返回类型 None

forward(*preds*, *data_samples*)

Compute FCENet loss.

参数

- **preds** (*list[dict]*) –A list of dict with keys of `cls_res`, `reg_res` corresponds to the classification result and regression result computed from the input tensor with the same index. They have the shapes of $(N, C_{cls,i}, H_i, W_i)$ and :math: *(N, C_{out,i}, H_i, W_i)*.
- **data_samples** (*list[TextDetDataSample]*) –The data samples.

返回

The dict for fcenet losses with loss_text, loss_center, loss_reg_x and loss_reg_y.

返回类型 dict

forward_single(*pred*, *gt*)

Compute loss for one feature level.

参数

- **pred** (*dict*) –A dict with keys `cls_res` and `reg_res` corresponds to the classification result and regression result from one feature level.
- **gt** (*Tensor*) –Ground truth for one feature level. Cls and reg targets are concatenated along the channel dimension.

返回 A list of losses for each feature level.

返回类型 list[Tensor]

get_targets(*data_samples*)

Generate loss targets for fcenet from data samples.

参数 **data_samples** (*list(TextDetDataSample)*) –Ground truth data samples.

返回

A tuple of three tensors from three different feature level as FCENet targets.

返回类型 tuple[Tensor]

class mmocr.models.textdet.module_losses.**PANModuleLoss**(*loss_text={'type': 'MaskedSquareDiceLoss'}*, *loss_kernel={'type': 'MaskedSquareDiceLoss'}*, *loss_embedding={'type': 'PANEmbLossV1'}*, *weight_text=1.0*, *weight_kernel=0.5*, *weight_embedding=0.25*, *ohem_ratio=3*, *shrink_ratio=(1.0, 0.5)*, *max_shrink_dist=20*, *reduction='mean'*)

The class for implementing PANet loss. This was partially adapted from https://github.com/whai362/pan_pp.pytorch and https://github.com/WenmuZhou/PAN.pytorch.

PANet: Efficient and Accurate Arbitrary- Shaped Text Detection with Pixel Aggregation Network.

参数

- **loss_text** (*dict*) –dict(type='MaskedSquareDiceLoss').
- **loss_kernel** (*dict*) –dict(type='MaskedSquareDiceLoss').
- **loss_embedding** (*dict*) –dict(type='PANEmbLossV1').
- **weight_text** (*float*) –The weight of text loss. Defaults to 1.
- **weight_kernel** (*float*) –The weight of kernel loss. Defaults to 0.5.
- **weight_embedding** (*float*) –The weight of embedding loss. Defaults to 0.25.
- **ohem_ratio** (*float*) –The negative/positive ratio in ohem. Defaults to 3.
- **shrink_ratio** (*tuple[float]*) –The ratio of shrinking kernel. Defaults to (1.0, 0.5).
- **max_shrink_dist** (*int or float*) –The maximum shrinking distance. Defaults to 20.
- **reduction** (*str*) –The way to reduce the loss. Available options are "mean" and "sum". Defaults to 'mean'.

返回类型 None

forward(*preds*, *data_samples*)

Compute PAN loss.

参数

- **preds** (*dict*) –Raw predictions from model with shape (N, C, H, W).
- **data_samples** (*list[TextDetDataSample]*) –The data samples.

返回 The dict for pan losses with loss_text, loss_kernel, loss_aggregation and loss_discrimination.

返回类型 dict

get_targets(*data_samples*)

Generate the gt targets for PANet.

参数

- **results** (*dict*) –The input result dictionary.
- **data_samples** (*Sequence[mmocr.structures.textdet_data_sample.TextDetDataSample]*) –

返回 The output result dictionary.

返回类型 results (dict)

class mmocr.models.textdet.module_losses.**PSEModuleLoss**(*weight_text=0.7*, *weight_kernel=0.3*, *loss_text={'type': 'MaskedSquareDiceLoss'}*, *loss_kernel={'type': 'MaskedSquareDiceLoss'}*, *ohem_ratio=3*, *reduction='mean'*, *kernel_sample_type='adaptive'*, *shrink_ratio=(1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4)*, *max_shrink_dist=20*)

The class for implementing PSENet loss. This is partially adapted from https://github.com/whai362/PSENet.

PSENet: Shape Robust Text Detection with Progressive Scale Expansion Network.

参数

- **weight_text** (*float*) –The weight of text loss. Defaults to 0.7.
- **weight_kernel** (*float*) –The weight of text kernel. Defaults to 0.3.
- **loss_text** (*dict*) –Loss type for text. Defaults to dict('MaskedSquareDiceLoss').
- **loss_kernel** (*dict*) –Loss type for kernel. Defaults to dict('MaskedSquareDiceLoss').
- **ohem_ratio** (*int or float*) –The negative/positive ratio in ohem. Defaults to 3.
- **reduction** (*str*) –The way to reduce the loss. Defaults to 'mean'. Options are 'mean' and 'sum'.
- **kernel_sample_type** (*str*) –The way to sample kernel. Defaults to adaptive. Options are 'adaptive' and 'hard'.

- **shrink_ratio** (*tuple*) –The ratio for shirinking text instances. Defaults to (1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4).
- **max_shrink_dist** (*int or float*) –The maximum shrinking distance. Defaults to 20.

返回类型 None

forward(*preds*, *data_samples*)

Compute PSENet loss.

参数

- **preds** (*torch.Tensor*) –Raw predictions from model with shape (N, C, H, W).
- **data_samples** (*list[TextDetDataSample]*) –The data samples.

返回 The dict for pse losses with loss_text, loss_kernel, loss_aggregation and loss_discrimination.

返回类型 dict

class mmocr.models.textdet.module_losses.**SegBasedModuleLoss**

Base class for the module loss of segmentation-based text detection algorithms with some handy utilities.

返回类型 None

class mmocr.models.textdet.module_losses.**TextSnakeModuleLoss**(*ohem_ratio=3.0*, *downsample_ratio=1.0*, *orientation_thr=2.0*, *resample_step=4.0*, *center_region_shrink_ratio=0.3*, *loss_text={'eps': 1e-05, 'fallback_negative_num': 100, 'type': 'MaskedBalancedBCEWithLogitsLoss'}*, *loss_center={'type': 'MaskedBCEWithLogitsLoss'}*, *loss_radius={'type': 'MaskedSmoothL1Loss'}*, *loss_sin={'type': 'MaskedSmoothL1Loss'}*, *loss_cos={'type': 'MaskedSmoothL1Loss'}*)

The class for implementing TextSnake loss. This is partially adapted from https://github.com/princewang1994/TextSnake.pytorch.

TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

参数

- **ohem_ratio** (*float*) –The negative/positive ratio in ohem.
- **downsample_ratio** (*float*) –Downsample ratio. Defaults to 1.0. TODO: remove it.
- **orientation_thr** (*float*) –The threshold for distinguishing between head edge and tail edge among the horizontal and vertical edges of a quadrangle.
- **resample_step** (*float*) –The step of resampling.
- **center_region_shrink_ratio** (*float*) –The shrink ratio of text center.
- **loss_text** (*dict*) –The loss config used to calculate the text loss.
- **loss_center** (*dict*) –The loss config used to calculate the center loss.
- **loss_radius** (*dict*) –The loss config used to calculate the radius loss.
- **loss_sin** (*dict*) –The loss config used to calculate the sin loss.
- **loss_cos** (*dict*) –The loss config used to calculate the cos loss.

返回类型 None

forward(*preds*, *data_samples*)

参数

- **preds** (*Tensor*) –The prediction map of shape $(N, 5, H, W)$, where each dimension is the map of "text_region", "center_region", "sin_map", "cos_map", and "radius_map" respectively.
- **data_samples** (*list[TextDetDataSample]*) –The data samples.

返回 A loss dict with `loss_text`, `loss_center`, `loss_radius`, `loss_sin` and `loss_cos`.

返回类型 dict

get_targets(*data_samples*)

Generate loss targets from data samples.

参数 **data_samples** (*list(TextDetDataSample)*) –Ground truth data samples.

返回 tuple(gt_text_masks, gt_masks, gt_center_region_masks, gt_radius_maps, gt_sin_maps, gt_cos_maps): A tuple of six lists of ndarrays as the targets.

返回类型 Tuple

vector_angle(*vec1*, *vec2*)

Compute the angle between two vectors.

参数

- **vec1** (*numpy.ndarray*) –
- **vec2** (*numpy.ndarray*) –

返回类型 numpy.ndarray

vector_cos(*vec*)

Compute the cos of the angle between vector and x-axis.

参数 **vec** (*numpy.ndarray*) –

返回类型 float

vector_sin(*vec*)

Compute the sin of the angle between vector and x-axis.

参数 **vec** (*numpy.ndarray*) –

返回类型 float

vector_slope(*vec*)

Compute the slope of a vector.

参数 **vec** (*numpy.ndarray*) –

返回类型 float

40.6 Text Detection Data Preprocessors

class mmocr.models.textdet.data_preprocessors.**TextDetDataPreprocessor**(*mean=None*, *std=None*, *pad_size_divisor=1*, *pad_value=0*, *bgr_to_rgb=False*, *rgb_to_bgr=False*, *batch_augments=None*)

Image pre-processor for detection tasks.

Comparing with the mmengine.ImgDataPreprocessor,

1. It supports batch augmentations.

2. It will additionally append batch_input_shape and pad_shape to data_samples considering the object detection task.

It provides the data pre-processing as follows

- Collate and move data to the target device.

- Pad inputs to the maximum size of current batch with defined `pad_value`. The padding size can be divisible by a defined `pad_size_divisor`
- Stack inputs to batch_inputs.
- Convert inputs from bgr to rgb if the shape of input is (3, H, W).
- Normalize image with defined std and mean.
- Do batch augmentations during training.

参数

- **mean** (*Sequence[Number], optional*) –The pixel mean of R, G, B channels. Defaults to None.
- **std** (*Sequence[Number], optional*) –The pixel standard deviation of R, G, B channels. Defaults to None.
- **pad_size_divisor** (*int*) –The size of padded image should be divisible by `pad_size_divisor`. Defaults to 1.
- **pad_value** (*Number*) –The padded pixel value. Defaults to 0.
- **pad_mask** (*bool*) –Whether to pad instance masks. Defaults to False.
- **mask_pad_value** (*int*) –The padded pixel value for instance masks. Defaults to 0.
- **pad_seg** (*bool*) –Whether to pad semantic segmentation maps. Defaults to False.
- **seg_pad_value** (*int*) –The padded pixel value for semantic segmentation maps. Defaults to 255.
- **bgr_to_rgb** (*bool*) –whether to convert image from BGR to RGB. Defaults to False.
- **rgb_to_bgr** (*bool*) –whether to convert image from RGB to RGB. Defaults to False.
- **batch_augments** (*list[dict], optional*) –Batch-level augmentations

返回类型 None

forward(*data*, *training=False*)

Perform normalization、padding and bgr2rgb conversion based on `BaseDataPreprocessor`.

参数

- **data** (*dict*) –data sampled from dataloader.
- **training** (*bool*) –Whether to enable training time augmentation.

返回 Data in the same format as the model input.

返回类型 dict

40.7 Text Detection Postprocessors

class mmocr.models.textdet.postprocessors.**BaseTextDetPostProcessor**(*text_repr_type='poly'*, *rescale_fields=None*, *train_cfg=None*, *test_cfg=None*)

Base postprocessor for text detection models.

参数

- **text_repr_type** (*str*) –The boundary encoding type, 'poly' or 'quad'. Defaults to 'poly'.
- **rescale_fields** (*list[str], optional*) –The bbox/polygon field names to be rescaled. If None, no rescaling will be performed.
- **train_cfg** (*dict, optional*) –The parameters to be passed to `self.get_text_instances` in training. Defaults to None.
- **test_cfg** (*dict, optional*) –The parameters to be passed to `self.get_text_instances` in testing. Defaults to None.

返回类型 None

get_text_instances(*pred_results*, *data_sample*, ***kwargs*)

Get text instance predictions of one image.

参数

- **pred_result** (*tuple(Tensor)*) –Prediction results of an image.
- **data_sample** (TextDetDataSample) –Datasample of an image.
- ****kwargs** –Other parameters. Configurable via `__init__.train_cfg` and `__init__.test_cfg`.
- **pred_results** (*Union[torch.Tensor, List[torch.Tensor]]*) –

返回 A new DataSample with predictions filled in. The polygon/bbox results are usually saved in `TextDetDataSample.pred_instances.polygons` or `TextDetDataSample.pred_instances.bboxes`. The confidence scores are saved in `TextDetDataSample.pred_instances.scores`.

返回类型 *TextDetDataSample*

poly_nms(*polygons*, *scores*, *threshold*)

Non-maximum suppression for text detection.

参数

- **polygons** (*list[ndarray]*) –List of polygons.

- **scores** (*list[float]*) –List of scores.
- **threshold** (*float*) –Threshold for NMS.

返回

- keep_polys (list[ndarray]): List of preserved polygons after NMS.
- keep_scores (list[float]): List of preserved scores after NMS.

返回类型 tuple(keep_polys, keep_scores)

rescale(*results*, *scale_factor*)

Rescale results in `results.pred_instances` according to `scale_factor`, whose keys are defined in `self.rescale_fields`. Usually used to rescale bboxes and/or polygons.

参数

- **results** (`TextDetDataSample`) –The post-processed prediction results.
- **scale_factor** (*tuple(int)*) –(w_scale, h_scale)

返回 Prediction results with rescaled results.

返回类型 *TextDetDataSample*

split_results(*pred_results*)

Split batched tensor(s) along the first dimension pack split tensors into a list.

参数 **pred_results** (*tensor or list[tensor]*) –Raw result tensor(s) from detection head. Each tensor usually has the shape of (N, …)

返回

N tensors if pred_results is a tensor, or a list of N lists of tensors if `pred_results` is a list of tensors.

返回类型 list[tensor] or list[list[tensor]]

class mmocr.models.textdet.postprocessors.**DBPostprocessor**(*text_repr_type='poly'*, *rescale_fields=['polygons']*, *mask_thr=0.3*, *min_text_score=0.3*, *min_text_width=5*, *unclip_ratio=1.5*, *epsilon_ratio=0.01*, *max_candidates=3000*, ***kwargs*)

Decoding predictions of DbNet to instances. This is partially adapted from https://github.com/MhLiao/DB.

参数

- **text_repr_type** (*str*) –The boundary encoding type 'poly' or 'quad'. Defaults to 'poly'.
- **rescale_fields** (*list[str]*) –The bbox/polygon field names to be rescaled. If None, no rescaling will be performed. Defaults to ['polygons'].
- **mask_thr** (*float*) –The mask threshold value for binarization. Defaults to 0.3.
- **min_text_score** (*float*) –The threshold value for converting binary map to shrink text regions. Defaults to 0.3.
- **min_text_width** (*int*) –The minimum width of boundary polygon/box predicted. Defaults to 5.
- **unclip_ratio** (*float*) –The unclip ratio for text regions dilation. Defaults to 1.5.
- **epsilon_ratio** (*float*) –The epsilon ratio for approximation accuracy. Defaults to 0.01.
- **max_candidates** (*int*) –The maximum candidate number. Defaults to 3000.

返回类型 None

get_text_instances(*prob_map*, *data_sample*)

Get text instance predictions of one image.

参数

- **pred_result** (*Tensor*) –DBNet's output `prob_map` of shape (H, W).
- **data_sample** (`TextDetDataSample`) –Datasample of an image.
- **prob_map** (*torch.Tensor*) –

返回 A new DataSample with predictions filled in. Polygons and results are saved in `TextDetDataSample.pred_instances.polygons`. The confidence scores are saved in `TextDetDataSample.pred_instances.scores`.

返回类型 *TextDetDataSample*

class mmocr.models.textdet.postprocessors.**DRRGPostprocessor**(*link_thr=0.8*, *edge_len_thr=50.0*, *rescale_fields=['polygons']*, ***kwargs*)

Merge text components and construct boundaries of text instances.

参数

- **link_thr** (*float*) –The edge score threshold. Defaults to 0.8.
- **edge_len_thr** (*int or float*) –The edge length threshold. Defaults to 50.
- **rescale_fields** (*list[str]*) –The bbox/polygon field names to be rescaled. If None, no rescaling will be performed. Defaults to [polygons'].

返回类型 None

get_text_instances(*pred_results*, *data_sample*)

Get text instance predictions of one image.

参数

- **pred_result** (*tuple(ndarray, ndarray, ndarray)*) –Prediction results edge, score and text_comps. Each of shape $(N_{edges}, 2)$, $(N_{edges},)$ and $(M, 9)$, respectively.
- **data_sample** (TextDetDataSample) –Datasample of an image.
- **pred_results** (*Tuple[numpy.ndarray, numpy.ndarray, numpy.ndarray]*) –

返回 The original dataSample with predictions filled in. Polygons and results are saved in `TextDetDataSample.pred_instances.polygons`. The confidence scores are saved in `TextDetDataSample.pred_instances.scores`.

返回类型 *TextDetDataSample*

split_results(*pred_results*)

Split batched elements in pred_results along the first dimension into `batch_num` sub-elements and regather them into a list of dicts.

However, DRRG only outputs one batch at inference time, so this function is a no-op.

参数 **pred_results** (*Tuple[numpy.ndarray, numpy.ndarray, numpy.ndarray]*) –

返回类型 List[Tuple]

class mmocr.models.textdet.postprocessors.**FCEPostprocessor**(*fourier_degree*, *num_reconstr_points*, *rescale_fields=['polygons']*, *scales=[8, 16, 32]*, *text_repr_type='poly'*, *alpha=1.0*, *beta=2.0*, *score_thr=0.3*, *nms_thr=0.1*, ***kwargs*)

Decoding predictions of FCENet to instances.

参数

- **fourier_degree** (*int*) –The maximum Fourier transform degree k.
- **num_reconstr_points** (*int*) –The points number of the polygon reconstructed from predicted Fourier coefficients.

- **rescale_fields** (*list[str]*) –The bbox/polygon field names to be rescaled. If None, no rescaling will be performed. Defaults to ['polygons'].
- **scales** (*list[int]*) –The down-sample scale of each layer. Defaults to [8, 16, 32].
- **text_repr_type** (*str*) –

  Boundary encoding type 'poly' or 'quad'. Defaults to 'poly'.

  alpha (float): The parameter to calculate final scores $Score_{final} = (Score^{a}_{textregion}lpha) * (Score^{b}_{textcenter_{r}egion}eta)$. Defaults to 1.0.
- **beta** (*float*) –The parameter to calculate final score. Defaults to 2.0.
- **score_thr** (*float*) –The threshold used to filter out the final candidates.Defaults to 0.3.
- **nms_thr** (*float*) –The threshold of nms. Defaults to 0.1.
- **alpha** (*float*) –

返回类型 None

get_text_instances(*pred_results*, *data_sample*)

Get text instance predictions of one image.

参数

- **pred_results** (*List[dict]*) –A list of dict with keys of `cls_res`, `reg_res` corresponding to the classification result and regression result computed from the input tensor with the same index. They have the shapes of $(N, C_{cls,i}, H_i, W_i)$ and $(N, C_{out,i}, H_i, W_i)$.
- **data_sample** (`TextDetDataSample`) –Datasample of an image.

返回 A new DataSample with predictions filled in. Polygons and results are saved in `TextDetDataSample.pred_instances.polygons`. The confidence scores are saved in `TextDetDataSample.pred_instances.scores`.

返回类型 *TextDetDataSample*

split_results(*pred_results*)

Split batched elements in pred_results along the first dimension into `batch_num` sub-elements and regather them into a list of dicts.

参数 **pred_results** (*list[dict]*) –A list of dict with keys of `cls_res`, `reg_res` corresponding to the classification result and regression result computed from the input tensor with the same index. They have the shapes of $(N, C_{cls,i}, H_i, W_i)$ and $(N, C_{out,i}, H_i, W_i)$.

返回 N lists. Each list contains three dicts from different feature level.

返回类型 list[list[dict]]

class mmocr.models.textdet.postprocessors.**PANPostprocessor**(*text_repr_type='poly'*, *score_threshold=0.3*, *rescale_fields=['polygons']*, *min_text_confidence=0.5*, *min_kernel_confidence=0.5*, *distance_threshold=3.0*, *min_text_area=16*, *downsample_ratio=0.25*)

Convert scores to quadrangles via post processing in PANet. This is partially adapted from https://github.com/WenmuZhou/PAN.pytorch.

参数

- **text_repr_type** (*str*) –The boundary encoding type 'poly' or 'quad'. Defaults to 'poly'.
- **score_threshold** (*float*) –The minimal text score. Defaults to 0.3.
- **rescale_fields** (*list[str]*) –The bbox/polygon field names to be rescaled. If None, no rescaling will be performed. Defaults to ['polygons'].
- **min_text_confidence** (*float*) –The minimal text confidence. Defaults to 0.5.
- **min_kernel_confidence** (*float*) –The minimal kernel confidence. Defaults to 0.5.
- **distance_threshold** (*float*) –The minimal distance between the point to mean of text kernel. Defaults to 3.0.
- **min_text_area** (*int*) –The minimal text instance region area. Defaults to 16.
- **downsample_ratio** (*float*) –Downsample ratio. Defaults to 0.25.

返回类型 None

get_text_instances(*pred_results*, *data_sample*, ***kwargs*)

Get text instance predictions of one image.

参数

- **pred_result** (*torch.Tensor*) –Prediction results of an image which is a tensor of shape (N, H, W).
- **data_sample** (TextDetDataSample) –Datasample of an image.
- **pred_results** (*torch.Tensor*) –

返回 A new DataSample with predictions filled in. Polygons and results are saved in `TextDetDataSample.pred_instances.polygons`. The confidence scores are saved in `TextDetDataSample.pred_instances.scores`.

返回类型 *TextDetDataSample*

split_results(*pred_results*)

Split the prediction results into text score and kernel score.

参数 **pred_results** (*torch.Tensor*) –The prediction results.

返回 The text score and kernel score.

返回类型 List[torch.Tensor]

class mmocr.models.textdet.postprocessors.**PSEPostprocessor**(*text_repr_type='poly'*, *rescale_fields=['polygons']*, *min_kernel_confidence=0.5*, *score_threshold=0.3*, *min_kernel_area=0*, *min_text_area=16*, *downsample_ratio=0.25*)

Decoding predictions of PSENet to instances. This is partially adapted from https://github.com/whai362/PSENet.

参数

- **text_repr_type** (*str*) –The boundary encoding type 'poly' or 'quad'. Defaults to 'poly'.
- **rescale_fields** (*list[str]*) –The bbox/polygon field names to be rescaled. If None, no rescaling will be performed. Defaults to ['polygons'].
- **min_kernel_confidence** (*float*) –The minimal kernel confidence. Defaults to 0.5.
- **score_threshold** (*float*) –The minimal text average confidence. Defaults to 0.3.
- **min_kernel_area** (*int*) –The minimal text kernel area. Defaults to 0.
- **min_text_area** (*int*) –The minimal text instance region area. Defaults to 16.
- **downsample_ratio** (*float*) –Downsample ratio. Defaults to 0.25.

返回类型 None

get_text_instances(*pred_results*, *data_sample*, ***kwargs*)

参数

- **pred_result** (*torch.Tensor*) –Prediction results of an image which is a tensor of shape (N, H, W).
- **data_sample** (TextDetDataSample) –Datasample of an image.
- **pred_results** (*torch.Tensor*) –

返回 A new DataSample with predictions filled in. Polygons and results are saved in TextDetDataSample.pred_instances.polygons. The confidence scores are

saved in TextDetDataSample.pred_instances.scores.

返回类型 *TextDetDataSample*

class mmocr.models.textdet.postprocessors.**TextSnakePostprocessor**(*text_repr_type='poly'*, *min_text_region_confidence=0.6*, *min_center_region_confidence=0.2*, *min_center_area=30*, *disk_overlap_thr=0.03*, *radius_shrink_ratio=1.03*, *rescale_fields=['polygons']*, ***kwargs*)

Decoding predictions of TextSnake to instances. This was partially adapted from https://github.com/princewang1994/TextSnake.pytorch.

参数

- **text_repr_type** (*str*) –The boundary encoding type 'poly' or 'quad'.
- **min_text_region_confidence** (*float*) –The confidence threshold of text region in TextSnake.
- **min_center_region_confidence** (*float*) –The confidence threshold of text center region in TextSnake.
- **min_center_area** (*int*) –The minimal text center region area.
- **disk_overlap_thr** (*float*) –The radius overlap threshold for merging disks.
- **radius_shrink_ratio** (*float*) –The shrink ratio of ordered disks radii.
- **rescale_fields** (*list[str], optional*) –The bbox/polygon field names to be rescaled. If None, no rescaling will be performed.

返回类型 None

get_text_instances(*pred_results*, *data_sample*)

参数

- **pred_results** (*torch.Tensor*) –Prediction map with shape (C, H, W).
- **data_sample** (TextDetDataSample) –Datasample of an image.

返回 The instance boundary and its confidence.

返回类型 list[list[float]]

split_results(*pred_results*)

Split the prediction results into text score and kernel score.

参数 **pred_results** (*torch.Tensor*) –The prediction results.

返回 The text score and kernel score.

返回类型 List[torch.Tensor]

40.8 Text Recognition Recognizer

class mmocr.models.textrecog.recognizers.**ABINet**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *data_preprocessor=None*, *init_cfg=None*)

Implementation of 'Read Like Humans: Autonomous, Bidirectional and Iterative LanguageModeling for Scene Text Recognition.

<https://arxiv.org/pdf/2103.06495.pdf>'_

参数

- **preprocessor** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **backbone** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **encoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **decoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **data_preprocessor** (*Union[mmengine.config.config.ConfigDict, Dict]*) –
- **init_cfg** (*Union[Dict, List[Dict]]*) –

返回类型 None

class mmocr.models.textrecog.recognizers.**BaseRecognizer**(*data_preprocessor=None*, *init_cfg=None*)

Base class for recognizer.

参数

- **data_preprocessor** (*dict or ConfigDict, optional*) –The pre-process config of BaseDataPreprocessor. it usually includes, pad_size_divisor, pad_value, mean and std.
- **init_cfg** (*dict or ConfigDict or List[dict], optional*) –the config to control the initialization. Defaults to None.

abstract extract_feat(*inputs*)

Extract features from images.

参数 **inputs** (*torch.Tensor*) –

返回类型 torch.Tensor

forward(*inputs*, *data_samples=None*, *mode='tensor'*, ***kwargs*)

The unified entry for a forward process in both training and test.

The method should accept three modes: "tensor", "predict" and "loss":

- "tensor": Forward the whole network and return tensor or tuple of

tensor without any post-processing, same as a common nn.Module. - "predict": Forward and return the predictions, which are fully processed to a list of `DetDataSample`. - "loss": Forward and return a dict of losses according to the given inputs and data samples.

Note that this method doesn't handle neither back propagation nor optimizer updating, which are done in the `train_step()`.

参数

- **inputs** (*torch.Tensor*) –The input tensor with shape (N, C, ···) in general.
- **data_samples** (list[`DetDataSample`], optional) –The annotation data of every samples. Defaults to None.
- **mode** (*str*) –Return what kind of value. Defaults to 'tensor'.

返回

The return type depends on `mode`.

- If `mode="tensor"`, return a tensor or a tuple of tensor.
- If `mode="predict"`, return a list of `DetDataSample`.
- If `mode="loss"`, return a dict of tensor.

返回类型 Union[Dict[str, torch.Tensor], List[*mmocr.structures.textrecog_data_sample.TextRecogDataSample*], Tuple[torch.Tensor], torch.Tensor]

abstract loss(*inputs*, *data_samples*, ***kwargs*)

Calculate losses from a batch of inputs and data samples.

参数

- **inputs** (*torch.Tensor*) –
- **data_samples** (*List[*`mmocr.structures.textrecog_data_sample.TextRecogDataSample`*]*) –

返回类型 Union[dict, tuple]

abstract predict(*inputs*, *data_samples*, ***kwargs*)

Predict results from a batch of inputs and data samples with post- processing.

参数

- **inputs** (*torch.Tensor*) –
- **data_samples** (*List[mmocr.structures.textrecog_data_sample.TextRecogDataSample]*) –

返回类型 List[*mmocr.structures.textrecog_data_sample.TextRecogDataSample*]

property with_backbone

whether the recognizer has a backbone

Type bool

property with_decoder

whether the recognizer has a decoder

Type bool

property with_encoder

whether the recognizer has an encoder

Type bool

property with_preprocessor

whether the recognizer has a preprocessor

Type bool

class mmocr.models.textrecog.recognizers.**CRNN**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *data_preprocessor=None*, *init_cfg=None*)

CTC-loss based recognizer.

参数

- **preprocessor** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **backbone** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **encoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **decoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **data_preprocessor** (*Union[mmengine.config.config.ConfigDict, Dict]*) –

- **init_cfg** (*Union[Dict, List[Dict]]*) –

返回类型 None

class mmocr.models.textrecog.recognizers.**EncoderDecoderRecognizer**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *data_preprocessor=None*, *init_cfg=None*)

Base class for encode-decode recognizer.

参数

- **preprocessor** (*dict, optional*) –Config dict for preprocessor. Defaults to None.
- **backbone** (*dict, optional*) –Backbone config. Defaults to None.
- **encoder** (*dict, optional*) –Encoder config. If None, the output from backbone will be directly fed into `decoder`. Defaults to None.
- **decoder** (*dict, optional*) –Decoder config. Defaults to None.
- **data_preprocessor** (*dict, optional*) –Model preprocessing config for processing the input image data. Keys allowed are "to_rgb"(bool), "pad_size_divisor"(int), "pad_value"(int or float), "mean"(int or float) and "std"(int or float). Preprcessing order: 1. to rgb; 2. normalization 3. pad. Defaults to None.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

extract_feat(*inputs*)

Directly extract features from the backbone.

参数 **inputs** (*torch.Tensor*) –

返回类型 torch.Tensor

loss(*inputs*, *data_samples*, ***kwargs*)

Calculate losses from a batch of inputs and data samples. :param inputs: Input images of shape (N, C, H, W).

Typically these should be mean centered and std scaled.

参数

- **data_samples** (*list[TextRecogDataSample]*) –A list of N datasamples, containing meta information and gold annotations for each of the images.
- **inputs** (*tensor*) –

返回 A dictionary of loss components.

返回类型 dict[str, tensor]

predict(*inputs*, *data_samples*, ***kwargs*)

Predict results from a batch of inputs and data samples with post- processing.

参数

- **inputs** (*torch.Tensor*) –Image input tensor.
- **data_samples** (*list[TextRecogDataSample]*) –A list of N datasamples, containing meta information and gold annotations for each of the images.

返回 A list of N datasamples of prediction results. Results are stored in `pred_text`.

返回类型 list[*TextRecogDataSample*]

class mmocr.models.textrecog.recognizers.**MASTER**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *data_preprocessor=None*, *init_cfg=None*)

Implementation of MASTER

参数

- **preprocessor** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **backbone** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **encoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **decoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **data_preprocessor** (*Union[mmengine.config.config.ConfigDict, Dict]*) –
- **init_cfg** (*Union[Dict, List[Dict]]*) –

返回类型 None

class mmocr.models.textrecog.recognizers.**NRTR**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *data_preprocessor=None*, *init_cfg=None*)

Implementation of NRTR

参数

- **preprocessor** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –

- **backbone** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **encoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **decoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **data_preprocessor** (*Union[mmengine.config.config.ConfigDict, Dict]*) –
- **init_cfg** (*Union[Dict, List[Dict]]*) –

返回类型 None

class mmocr.models.textrecog.recognizers.**RobustScanner**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *data_preprocessor=None*, *init_cfg=None*)

Implementation of 'RobustScanner.

<https://arxiv.org/pdf/2007.07542.pdf>

参数

- **preprocessor** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **backbone** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **encoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **decoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **data_preprocessor** (*Union[mmengine.config.config.ConfigDict, Dict]*) –
- **init_cfg** (*Union[Dict, List[Dict]]*) –

返回类型 None

class mmocr.models.textrecog.recognizers.**SARNet**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *data_preprocessor=None*, *init_cfg=None*)

Implementation of SAR

参数

- **preprocessor** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **backbone** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **encoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **decoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **data_preprocessor** (*Union[mmengine.config.config.ConfigDict, Dict]*) –
- **init_cfg** (*Union[Dict, List[Dict]]*) –

返回类型 None

class mmocr.models.textrecog.recognizers.**SATRN**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *data_preprocessor=None*, *init_cfg=None*)

Implementation of SATRN

参数

- **preprocessor** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **backbone** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **encoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **decoder** (*Optional[Union[mmengine.config.config.ConfigDict, Dict]]*) –
- **data_preprocessor** (*Union[mmengine.config.config.ConfigDict, Dict]*) –
- **init_cfg** (*Union[Dict, List[Dict]]*) –

返回类型 None

40.9 Text Recognition Backbones

class mmocr.models.textrecog.backbones.**MiniVGG**(*leaky_relu=True*, *input_channels=3*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

A mini VGG backbone for text recognition, modified from 'VGG-VeryDeep.

<https://arxiv.org/pdf/1409.1556.pdf>'_

参数

- **leaky_relu** (*bool*) –Use leakyRelu or not.
- **input_channels** (*int*) –Number of channels of input image tensor.

forward(*x*)

参数 **x** (*Tensor*) –Images of shape (N, C, H, W).

返回 The feature Tensor of shape $(N, 512, H/32, (W/4+1))$.

返回类型 Tensor

class mmocr.models.textrecog.backbones.**MobileNetV2**(*pooling_layers=[3, 4, 5]*, *init_cfg=None*)

See mmdet.models.backbones.MobileNetV2 for details.

参数

- **pooling_layers** (*list*) –List of indices of pooling layers.
- **init_cfg** (*InitConfigType, optional*) –Initialization config dict.

返回类型 None

forward(*x*)

Forward function.

参数 **x** (*torch.Tensor*) –

返回类型 torch.Tensor

class mmocr.models.textrecog.backbones.**NRTRModalityTransform**(*in_channels=3*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

Modality transform in NRTR.

参数

- **in_channels** (*int*) –Input channel of image. Defaults to 3.

- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

forward(*x*)

Backbone forward.

参数 **x** (*torch.Tensor*) –Image tensor of shape (N, C, W, H). W, H is the width and height of image.

返回 Output tensor.

返回类型 Tensor

class mmocr.models.textrecog.backbones.**ResNet**(*in_channels*, *stem_channels*, *block_cfgs*, *arch_layers*, *arch_channels*, *strides*, *out_indices=None*, *plugins=None*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}, {'type': 'Constant', 'val': 1, 'layer': 'BatchNorm2d'}]*)

参数

- **in_channels** (*int*) –Number of channels of input image tensor.
- **stem_channels** (*list[int]*) –List of channels in each stem layer. E.g., [64, 128] stands for 64 and 128 channels in the first and second stem layers.
- **block_cfgs** (*dict*) –Configs of block
- **arch_layers** (*list[int]*) –List of Block number for each stage.
- **arch_channels** (*list[int]*) –List of channels for each stage.
- **strides** (*Sequence[int] or Sequence[tuple]*) –Strides of the first block of each stage.
- **out_indices** (*Sequence[int], optional*) –Indices of output stages. If not specified, only the last stage will be returned.
- **plugins** (*dict, optional*) –Configs of stage plugins
- **init_cfg** (*dict or list[dict], optional*) –Initialization config dict.

forward(*x*)

Args: x (Tensor): Image tensor of shape $(N, 3, H, W)$.

返回 Feature tensor. It can be a list of feature outputs at specific layers if `out_indices` is specified.

返回类型 Tensor or list[Tensor]

参数 **x** (*torch.Tensor*) –

forward_plugin(*x*, *plugin_name*)

Forward tensor through plugin.

参数

- **x** (*torch.Tensor*) –Input tensor.
- **plugin_name** (*list[str]*) –Name of plugins.

返回 Output tensor.

返回类型 torch.Tensor

class mmocr.models.textrecog.backbones.**ResNet31OCR**(*base_channels=3*, *layers=[1, 2, 5, 3]*, *channels=[64, 128, 256, 256, 512, 512, 512]*, *out_indices=None*, *stage4_pool_cfg={'kernel_size': (2, 1), 'stride': (2, 1)}*, *last_stage_pool=False*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

Implement ResNet backbone for text recognition, modified from ResNet

参数

- **base_channels** (*int*) –Number of channels of input image tensor.
- **layers** (*list[int]*) –List of BasicBlock number for each stage.
- **channels** (*list[int]*) –List of out_channels of Conv2d layer.
- **out_indices** (*None | Sequence[int]*) –Indices of output stages.
- **stage4_pool_cfg** (*dict*) –Dictionary to construct and configure pooling layer in stage 4.
- **last_stage_pool** (*bool*) –If True, add *MaxPool2d* layer to last stage.

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.

---

class mmocr.models.textrecog.backbones.**ResNetABI**(*in_channels=3*, *stem_channels=32*, *base_channels=32*, *arch_settings=[3, 4, 6, 6, 3]*, *strides=[2, 1, 2, 1, 1]*, *out_indices=None*, *last_stage_pool=False*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}, {'type': 'Constant', 'val': 1, 'layer': 'BatchNorm2d'}]*)

Implement ResNet backbone for text recognition, modified from 'ResNet.

<https://arxiv.org/pdf/1512.03385.pdf>'_ and https://github.com/FangShancheng/ABINet

参数

- **in_channels** (*int*) –Number of channels of input image tensor.
- **stem_channels** (*int*) –Number of stem channels.
- **base_channels** (*int*) –Number of base channels.
- **arch_settings** (*list[int]*) –List of BasicBlock number for each stage.
- **strides** (*Sequence[int]*) –Strides of the first block of each stage.
- **out_indices** (*None | Sequence[int]*) –Indices of output stages. If not specified, only the last stage will be returned.
- **last_stage_pool** (*bool*) –If True, add *MaxPool2d* layer to last stage.

forward(*x*)

参数 **x** (*Tensor*) –Image tensor of shape $(N, 3, H, W)$.

返回 Feature tensor. Its shape depends on ResNetABI' s config. It can be a list of feature outputs at specific layers if `out_indices` is specified.

返回类型 Tensor or list[Tensor]

class mmocr.models.textrecog.backbones.**ShallowCNN**(*input_channels=1*, *hidden_dim=512*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

Implement Shallow CNN block for SATRN.

SATRN: On Recognizing Texts of Arbitrary Shapes with 2D Self-Attention.

参数

- **input_channels** (*int*) –Number of channels of input image tensor D_i. Defaults to 1.
- **hidden_dim** (*int*) –Size of hidden layers of the model D_m. Defaults to 512.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

forward(*x*)

参数 **x** (*Tensor*) –Input image feature (N, D_i, H, W).

返回 A tensor of shape $(N, D_m, H/4, W/4)$.

返回类型 Tensor

40.10 Text Recognition Data Preprocessors

class mmocr.models.textrecog.data_preprocessors.**TextRecogDataPreprocessor**(*mean=None*, *std=None*, *pad_size_divisor=1*, *pad_value=0*, *bgr_to_rgb=False*, *rgb_to_bgr=False*, *batch_augments=None*)

Image pre-processor for recognition tasks.

Comparing with the `mmengine.ImgDataPreprocessor`,

1. It supports batch augmentations.

2. It will additionally append batch_input_shape and valid_ratio to data_samples considering the object recognition task.

It provides the data pre-processing as follows

- Collate and move data to the target device.
- Pad inputs to the maximum size of current batch with defined `pad_value`. The padding size can be divisible by a defined `pad_size_divisor`
- Stack inputs to inputs.
- Convert inputs from bgr to rgb if the shape of input is (3, H, W).
- Normalize image with defined std and mean.
- Do batch augmentations during training.

参数

- **mean** (*Sequence[Number], optional*) –The pixel mean of R, G, B channels. Defaults to None.
- **std** (*Sequence[Number], optional*) –The pixel standard deviation of R, G, B channels. Defaults to None.

- **pad_size_divisor** (*int*) –The size of padded image should be divisible by `pad_size_divisor`. Defaults to 1.
- **pad_value** (*Number*) –The padded pixel value. Defaults to 0.
- **bgr_to_rgb** (*bool*) –whether to convert image from BGR to RGB. Defaults to False.
- **rgb_to_bgr** (*bool*) –whether to convert image from RGB to RGB. Defaults to False.
- **batch_augments** (*list[dict], optional*) –Batch-level augmentations

返回类型 None

forward(*data*, *training=False*)

Perform normalization、padding and bgr2rgb conversion based on `BaseDataPreprocessor`.

参数

- **data** (*dict*) –Data sampled from dataloader.
- **training** (*bool*) –Whether to enable training time augmentation.

返回 Data in the same format as the model input.

返回类型 dict

40.11 Text Recognition Layers

class mmocr.models.textrecog.layers.**Adaptive2DPositionalEncoding**(*d_hid=512*, *n_height=100*, *n_width=100*, *dropout=0.1*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}]*)

Implement Adaptive 2D positional encoder for SATRN, see ‘SATRN.

<https://arxiv.org/abs/1910.04396>‘_ Modified from https://github.com/Media-Smart/vedastr Licensed under the Apache License, Version 2.0 (the “License”);

参数

- **d_hid** (*int*) –Dimensions of hidden layer. Defaults to 512.
- **n_height** (*int*) –Max height of the 2D feature output. Defaults to 100.
- **n_width** (*int*) –Max width of the 2D feature output. Defaults to 100.
- **dropout** (*float*) –Dropout rate. Defaults to 0.1.

- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to [dict(type=’Xavier’, layer=’Conv2d’)]

返回类型 None

forward(*x*)

Forward propagation of Locality Aware Feedforward module.

参数 **x** (*Tensor*) –Feature tensor.

返回 Feature tensor after Locality Aware Feedforward.

返回类型 Tensor

class mmocr.models.textrecog.layers.**BasicBlock**(*inplanes*, *planes*, *stride=1*, *downsample=None*, *use_conv1x1=False*, *plugins=None*)

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.

---

make_block_plugins(*in_channels*, *plugins*)

make plugins for block.

参数

- **in_channels** (*int*) –Input channels of plugin.
- **plugins** (*list[dict]*) –List of plugins cfg to build.

返回 List of the names of plugin.

返回类型 list[str]

class mmocr.models.textrecog.layers.**BidirectionalLSTM**(*nIn*, *nHidden*, *nOut*)

forward(*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while

the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**Bottleneck**(*inplanes*, *planes*, *stride=1*, *downsample=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.

---

class mmocr.models.textrecog.layers.**DotProductAttentionLayer**(*dim_model=None*)

forward(*query*, *key*, *value*, *mask=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**PositionAwareLayer**(*dim_model*, *rnn_layers=2*)

forward(*img_feature*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**RobustScannerFusionLayer**(*dim_model*, *dim=- 1*, *init_cfg=None*)

forward(*x0*, *x1*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解： Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**SATRNEncoderLayer**(*d_model=512*, *d_inner=512*, *n_head=8*, *d_k=64*, *d_v=64*, *dropout=0.1*, *qkv_bias=False*, *init_cfg=None*)

Implement encoder layer for SATRN, see ‘SATRN.

<https://arxiv.org/abs/1910.04396>‘_.

参数

- **d_model** (*int*) –Dimension D_m of the input from previous model. Defaults to 512.
- **d_inner** (*int*) –Hidden dimension of feedforward layers. Defaults to 256.
- **n_head** (*int*) –Number of parallel attention heads. Defaults to 8.
- **d_k** (*int*) –Dimension of the key vector. Defaults to 64.
- **d_v** (*int*) –Dimension of the value vector. Defaults to 64.
- **dropout** (*float*) –Dropout rate. Defaults to 0.1.
- **qkv_bias** (*bool*) –Whether to use bias. Defaults to False.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

forward(*x*, *h*, *w*, *mask=None*)

Forward propagation of encoder.

参数

- **x** (*Tensor*) –Feature tensor of shape $(N, h * w, D_m)$.
- **h** (*int*) –Height of the original feature.
- **w** (*int*) –Width of the original feature.
- **mask** (*Tensor, optional*) –Mask used for masked multi-head attention. Defaults to None.

返回 A tensor of shape $(N, h * w, D_m)$.

返回类型 Tensor

40.12 Text Recognition Plugins

class mmocr.models.textrecog.plugins.**GCAModule**(*in_channels*, *ratio*, *n_head*, *pooling_type='att'*, *scale_attn=False*, *fusion_type='channel_add'*, ***kwargs*)

GCAModule in MASTER.

参数

- **in_channels** (*int*) –Channels of input tensor.
- **ratio** (*float*) –Scale ratio of in_channels.
- **n_head** (*int*) –Numbers of attention head.
- **pooling_type** (*str*) –Spatial pooling type. Options are [`avg`, `att`].
- **scale_attn** (*bool*) –Whether to scale the attention map. Defaults to False.
- **fusion_type** (*str*) –Fusion type of input and context. Options are [`channel_add`, `channel_mul`, `channel_concat`].

返回类型 None

forward(*x*)

Forward function.

参数 **x** (*Tensor*) –Input feature map.

返回 Output tensor after GCAModule.

返回类型 Tensor

spatial_pool(*x*)

Spatial pooling function.

参数 **x** (*Tensor*) –Input feature map.

返回 Output tensor after spatial pooling.

返回类型 Tensor

class mmocr.models.textrecog.plugins.**Maxpool2d**(*kernel_size*, *stride*, *padding=0*, ***kwargs*)

A wrapper around nn.Maxpool2d().

参数

- **kernel_size** (*int or tuple(int)*) –Kernel size for max pooling layer
- **stride** (*int or tuple(int)*) –Stride for max pooling layer

- **padding** (*int or tuple(int)*) –Padding for pooling layer

返回类型 None

forward(*x*)

Forward function. :param x: Input feature map. :type x: Tensor

返回 Output tensor after Maxpooling layer.

返回类型 Tensor

40.13 Text Recognition Encoders

class mmocr.models.textrecog.encoders.**ABIEncoder**(*n_layers=2*, *n_head=8*, *d_model=512*, *d_inner=2048*, *dropout=0.1*, *max_len=256*, *init_cfg=None*)

Implement transformer encoder for text recognition, modified from <*https://github.com/FangShancheng/ABINet*>.

参数

- **n_layers** (*int*) –Number of attention layers. Defaults to 2.
- **n_head** (*int*) –Number of parallel attention heads. Defaults to 8.
- **d_model** (*int*) –Dimension D_m of the input from previous model. Defaults to 512.
- **d_inner** (*int*) –Hidden dimension of feedforward layers. Defaults to 2048.
- **dropout** (*float*) –Dropout rate. Defaults to 0.1.
- **max_len** (*int*) –Maximum output sequence length T. Defaults to 8 * 32.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

forward(*feature*, *data_samples*)

参数

- **feature** (*Tensor*) –Feature tensor of shape (N, D_m, H, W).
- **data_samples** (*List[TextRecogDataSample]*) –List of data samples.

返回 Features of shape (N, D_m, H, W).

返回类型 Tensor

class mmocr.models.textrecog.encoders.**BaseEncoder**(*init_cfg=None*)

Base Encoder class for text recognition.

forward(*feat*, ***kwargs*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解： Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.encoders.**ChannelReductionEncoder**(*in_channels*, *out_channels*, *init_cfg={'layer': 'Conv2d', 'type': 'Xavier'}*)

Change the channel number with a one by one convoluational layer.

参数

- **in_channels** (*int*) –Number of input channels.
- **out_channels** (*int*) –Number of output channels.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to dict(type='Xavier', layer='Conv2d').

返回类型 None

forward(*feat*, *data_samples=None*)

参数

- **feat** (*Tensor*) –Image features with the shape of (N, C_{in}, H, W).
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing valid_ratio information. Defaults to None.

返回 A tensor of shape (N, C_{out}, H, W).

返回类型 Tensor

class mmocr.models.textrecog.encoders.**NRTREncoder**(*n_layers=6*, *n_head=8*, *d_k=64*, *d_v=64*, *d_model=512*, *d_inner=256*, *dropout=0.1*, *init_cfg=None*)

Transformer Encoder block with self attention mechanism.

参数

- **n_layers** (*int*) –The number of sub-encoder-layers in the encoder. Defaults to 6.
- **n_head** (*int*) –The number of heads in the multiheadattention models Defaults to 8.
- **d_k** (*int*) –Total number of features in key. Defaults to 64.

- **d_v** (*int*) –Total number of features in value. Defaults to 64.
- **d_model** (*int*) –The number of expected features in the decoder inputs. Defaults to 512.
- **d_inner** (*int*) –The dimension of the feedforward network model. Defaults to 256.
- **dropout** (*float*) –Dropout rate for MHSA and FFN. Defaults to 0.1.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

forward(*feat*, *data_samples=None*)

参数

- **feat** (*Tensor*) –Backbone output of shape (N, C, H, W).
- **data_samples** (*list[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing valid_ratio information. Defaults to None.

返回 The encoder output tensor. Shape (N, T, C).

返回类型 Tensor

class mmocr.models.textrecog.encoders.**SAREncoder**(*enc_bi_rnn=False*, *rnn_dropout=0.0*, *enc_gru=False*, *d_model=512*, *d_enc=512*, *mask=True*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*, ***kwargs*)

Implementation of encoder module in 'SAR.

<https://arxiv.org/abs/1811.00751>'_.

参数

- **enc_bi_rnn** (*bool*) –If True, use bidirectional RNN in encoder. Defaults to False.
- **rnn_dropout** (*float*) –Dropout probability of RNN layer in encoder. Defaults to 0.0.
- **enc_gru** (*bool*) –If True, use GRU, else LSTM in encoder. Defaults to False.
- **d_model** (*int*) –Dim D_i of channels from backbone. Defaults to 512.
- **d_enc** (*int*) –Dim D_m of encoder RNN layer. Defaults to 512.
- **mask** (*bool*) –If True, mask padding in RNN sequence. Defaults to True.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to [dict(type=' Xavier ' , layer=' Conv2d '), dict(type=' Uniform ' , layer=' BatchNorm2d ')].

返回类型 None

forward(*feat*, *data_samples=None*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing valid_ratio information. Defaults to None.

返回 A tensor of shape (N, D_m).

返回类型 Tensor

class mmocr.models.textrecog.encoders.**SATRNEncoder**(*n_layers=12*, *n_head=8*, *d_k=64*, *d_v=64*, *d_model=512*, *n_position=100*, *d_inner=256*, *dropout=0.1*, *init_cfg=None*)

Implement encoder for SATRN, see ‘SATRN.

<https://arxiv.org/abs/1910.04396>‘_.

参数

- **n_layers** (*int*) –Number of attention layers. Defaults to 12.
- **n_head** (*int*) –Number of parallel attention heads. Defaults to 8.
- **d_k** (*int*) –Dimension of the key vector. Defaults to 64.
- **d_v** (*int*) –Dimension of the value vector. Defaults to 64.
- **d_model** (*int*) –Dimension D_m of the input from previous model. Defaults to 512.
- **n_position** (*int*) –Length of the positional encoding vector. Must be greater than `max_seq_len`. Defaults to 100.
- **d_inner** (*int*) –Hidden dimension of feedforward layers. Defaults to 256.
- **dropout** (*float*) –Dropout rate. Defaults to 0.1.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

forward(*feat*, *data_samples=None*)

Forward propagation of encoder.

参数

- **feat** (*Tensor*) –Feature tensor of shape (N, D_m, H, W).
- **data_samples** (*list[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing *valid_ratio* information. Defaults to None.

返回 A tensor of shape (N, T, D_m).

返回类型 Tensor

40.14 Text Recognition Decoders

class mmocr.models.textrecog.decoders.**ABIFuser**(*dictionary*, *vision_decoder*, *language_decoder=None*, *d_model=512*, *num_iters=1*, *max_seq_len=40*, *module_loss=None*, *postprocessor=None*, *init_cfg=None*, ***kwargs*)

A special decoder responsible for mixing and aligning visual feature and linguistic feature. ABINet

参数

- **dictionary** (dict or Dictionary) –The config for *Dictionary* or the instance of *Dictionary*. The dictionary must have an end token.
- **vision_decoder** (*dict*) –The config for vision decoder.
- **language_decoder** (*dict, optional*) –The config for language decoder.
- **num_iters** (*int*) –Rounds of iterative correction. Defaults to 1.
- **d_model** (*int*) –Hidden size E of model. Defaults to 512.
- **max_seq_len** (*int*) –Maximum sequence length T. The sequence is usually generated from decoder. Defaults to 40.
- **module_loss** (*dict, optional*) –Config to build loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

forward_test(*feat*, *logits*, *data_samples=None*)

参数

- **feat** (*torch.Tensor, optional*) –Not required. Feature map placeholder. Defaults to None.
- **logits** (*Tensor*) –Raw language logitis. Shape (N, T, C).
- **data_samples** (*list[TextRecogDataSample], optional*) –Not required. DataSample placeholder. Defaults to None.

返回 Character probabilities. of shape $(N, self.max_seq_len, C)$ where C is `num_classes`.

返回类型 Tensor

forward_train(*feat=None*, *out_enc=None*, *data_samples=None*)

参数

- **feat** (*torch.Tensor, optional*) –Not required. Feature map placeholder. Defaults to None.
- **out_enc** (*Tensor*) –Raw language logitis. Shape (N, T, C). Defaults to None.
- **data_samples** (*list[TextRecogDataSample], optional*) –Not required. DataSample placeholder. Defaults to None.

返回

A dict with keys `out_enc`, `out_decs` and `out_fusers`.

- out_vis (dict): Dict from `self.vision_decoder` with keys `feature`, `logits` and `attn_scores`.
- out_langs (dict or list): Dict from `self.vision_decoder` with keys `feature`, `logits` if applicable, or an empty list otherwise.
- out_fusers (dict or list): Dict of fused visual and language features with keys `feature`, `logits` if applicable, or an empty list otherwise.

返回类型 Dict

fuse(*l_feature*, *v_feature*)

Mix and align visual feature and linguistic feature.

参数

- **l_feature** (*torch.Tensor*) –(N, T, E) where T is length, N is batch size and E is dim of model.
- **v_feature** (*torch.Tensor*) –(N, T, E) shape the same as l_feature.

返回 A dict with key `logits`. of shape (N, T, C) where N is batch size, T is length and C is the number of characters.

返回类型 dict

class mmocr.models.textrecog.decoders.**ABILanguageDecoder**(*dictionary*, *d_model=512*, *n_head=8*, *d_inner=2048*, *n_layers=4*, *dropout=0.1*, *detach_tokens=True*, *use_self_attn=False*, *max_seq_len=40*, *module_loss=None*, *postprocessor=None*, *init_cfg=None*, ***kwargs*)

Transformer-based language model responsible for spell correction. Implementation of language model of ABINet.

参数

- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*. The dictionary must have an end token.
- **d_model** (*int*) –Hidden size E of model. Defaults to 512.
- **n_head** (*int*) –Number of multi-attention heads.
- **d_inner** (*int*) –Hidden size of feedforward network model.
- **n_layers** (*int*) –The number of similar decoding layers.
- **dropout** (*float*) –Dropout rate.
- **detach_tokens** (*bool*) –Whether to block the gradient flow at input tokens.
- **use_self_attn** (*bool*) –If True, use self attention in decoder layers, otherwise cross attention will be used.
- **max_seq_len** (*int*) –Maximum sequence length T. The sequence is usually generated from decoder. Defaults to 40.
- **module_loss** (*dict, optional*) –Config to build loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

forward_test(*feat=None*, *logits=None*, *data_samples=None*)

参数

- **feat** (*torch.Tensor, optional*) –Not required. Feature map placeholder. Defaults to None.
- **logits** (*Tensor*) –Raw language logitis. Shape (N, T, C). Defaults to None.
- **data_samples** (*list[TextRecogDataSample], optional*) –Not required. DataSample placeholder. Defaults to None.

返回

A dict with keys `feature` and `logits`.

- feature (Tensor): Shape (N, T, E). Raw textual features for vision language aligner.
- logits (Tensor): Shape (N, T, C). The raw logits for characters after spell correction.

返回类型 Dict

forward_train(*feat=None*, *out_enc=None*, *data_samples=None*)

参数

- **feat** (*torch.Tensor, optional*) –Not required. Feature map placeholder. Defaults to None.
- **out_enc** (*torch.Tensor*) –Logits with shape (N, T, C). Defaults to None.
- **data_samples** (*list[TextRecogDataSample], optional*) –Not required. DataSample placeholder. Defaults to None.

返回

A dict with keys `feature` and `logits`.

- feature (Tensor): Shape (N, T, E). Raw textual features for vision language aligner.
- logits (Tensor): Shape (N, T, C). The raw logits for characters after spell correction.

返回类型 Dict

class mmocr.models.textrecog.decoders.**ABIVisionDecoder**(*dictionary*, *in_channels=512*, *num_channels=64*, *attn_height=8*, *attn_width=32*, *attn_mode='nearest'*, *module_loss=None*, *postprocessor=None*, *max_seq_len=40*, *init_cfg={'layer': 'Conv2d', 'type': 'Xavier'}*, ***kwargs*)

Converts visual features into text characters.

Implementation of VisionEncoder in ABINet.

参数

- **dictionary** (dict or Dictionary) –The config for *Dictionary* or the instance of *Dictionary*.
- **in_channels** (*int*) –Number of channels E of input vector. Defaults to 512.
- **num_channels** (*int*) –Number of channels of hidden vectors in mini U-Net. Defaults to 64.
- **attn_height** (*int*) –Height H of input image features. Defaults to 8.
- **attn_width** (*int*) –Width W of input image features. Defaults to 32.
- **attn_mode** (*str*) –Upsampling mode for `torch.nn.Upsample` in mini U-Net. Defaults to 'nearest'.
- **module_loss** (*dict, optional*) –Config to build loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **max_seq_len** (*int*) –Maximum sequence length. The sequence is usually generated from decoder. Defaults to 40.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to dict(type='Xavier', layer='Conv2d').

返回类型 None

forward_test(*feat=None*, *out_enc=None*, *data_samples=None*)

参数

- **feat** (*torch.Tensor, optional*) –Image features of shape (N, E, H, W). Defaults to None.
- **out_enc** (*torch.Tensor*) –Encoder output. Defaults to None.
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回

A dict with keys `feature`, `logits` and `attn_scores`.

- feature (Tensor): Shape (N, T, E). Raw visual features for language decoder.
- logits (Tensor): Shape (N, T, C). The raw logits for characters.
- attn_scores (Tensor): Shape (N, T, H, W). Intermediate result for vision-language aligner.

返回类型 dict

forward_train(*feat=None*, *out_enc=None*, *data_samples=None*)

参数

- **feat** (*Tensor, optional*) –Image features of shape (N, E, H, W). Defaults to None.
- **out_enc** (*torch.Tensor*) –Encoder output. Defaults to None.
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回

A dict with keys `feature`, `logits` and `attn_scores`.

- feature (Tensor): Shape (N, T, E). Raw visual features for language decoder.
- logits (Tensor): Shape (N, T, C). The raw logits for characters.
- attn_scores (Tensor): Shape (N, T, H, W). Intermediate result for vision-language aligner.

返回类型 dict

class mmocr.models.textrecog.decoders.**BaseDecoder**(*dictionary*, *module_loss=None*, *postprocessor=None*, *max_seq_len=40*, *init_cfg=None*)

Base decoder for text recognition, build the loss and postprocessor.

参数

- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*.
- **loss** (*dict, optional*) –Config to build loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **max_seq_len** (*int*) –Maximum sequence length. The sequence is usually generated from decoder. Defaults to 40.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.
- **module_loss** (*Optional[Dict]*) –

返回类型 None

forward(*feat=None*, *out_enc=None*, *data_samples=None*)

Decoder forward.

Args:

feat (Tensor, optional): Features from the backbone. Defaults to None.

out_enc (Tensor, optional): Features from the encoder. Defaults to None.

data_samples (list[TextRecogDataSample]): A list of N datasamples, containing meta information and gold annotations for each of the images. Defaults to None.

返回 Features from `decoder` forward.

返回类型 Tensor

参数

- **feat** (*Optional[torch.Tensor]*) –
- **out_enc** (*Optional[torch.Tensor]*) –
- **data_samples** (*Optional[Sequence[mmocr.structures.textrecog_data_sample.TextRecogDataSample]]*) –

forward_test(*feat=None*, *out_enc=None*, *data_samples=None*)

Forward for testing.

参数

- **feat** (*torch.Tensor, optional*) –The feature map from backbone of shape (N, E, H, W). Defaults to None.
- **out_enc** (*torch.Tensor, optional*) –Encoder output. Defaults to None.
- **data_samples** (*Sequence[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回类型 torch.Tensor

forward_train(*feat=None*, *out_enc=None*, *data_samples=None*)

Forward for training.

参数

- **feat** (*torch.Tensor, optional*) –The feature map from backbone of shape (N, E, H, W). Defaults to None.
- **out_enc** (*torch.Tensor, optional*) –Encoder output. Defaults to None.
- **data_samples** (*Sequence[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回类型 torch.Tensor

loss(*feat=None*, *out_enc=None*, *data_samples=None*)

Calculate losses from a batch of inputs and data samples.

参数

- **feat** (*Tensor, optional*) –Features from the backbone. Defaults to None.
- **out_enc** (*Tensor, optional*) –Features from the encoder. Defaults to None.
- **data_samples** (*list[TextRecogDataSample], optional*) –A list of N datasamples, containing meta information and gold annotations for each of the images. Defaults to None.

返回 A dictionary of loss components.

返回类型 dict[str, tensor]

predict(*feat=None*, *out_enc=None*, *data_samples=None*)

Perform forward propagation of the decoder and postprocessor.

参数

- **feat** (*Tensor, optional*) –Features from the backbone. Defaults to None.
- **out_enc** (*Tensor, optional*) –Features from the encoder. Defaults to None.
- **data_samples** (*list[TextRecogDataSample]*) –A list of N datasamples, containing meta information and gold annotations for each of the images. Defaults to None.

返回 A list of N datasamples of prediction results. Results are stored in `pred_text`.

返回类型 list[*TextRecogDataSample*]

class mmocr.models.textrecog.decoders.**CRNNDecoder**(*in_channels*, *dictionary*, *rnn_flag=False*, *module_loss=None*, *postprocessor=None*, *init_cfg={'layer': 'Conv2d', 'type': 'Xavier'}*, ***kwargs*)

Decoder for CRNN.

参数

- **in_channels** (*int*) –Number of input channels.
- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*.
- **rnn_flag** (*bool*) –Use RNN or CNN as the decoder. Defaults to False.
- **module_loss** (*dict, optional*) –Config to build module_loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

forward_test(*feat=None*, *out_enc=None*, *data_samples=None*)

参数

- **feat** (*Tensor*) –A Tensor of shape $(N, C, 1, W)$.
- **out_enc** (*torch.Tensor, optional*) –Encoder output. Defaults to None.
- **data_samples** (*list[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing `gt_text` information. Defaults to None.

返回 Character probabilities. of shape $(N, self.max_seq_len, C)$ where C is `num_classes`.

返回类型 Tensor

forward_train(*feat*, *out_enc=None*, *data_samples=None*)

参数

- **feat** (*Tensor*) –A Tensor of shape $(N, C, 1, W)$.
- **out_enc** (*torch.Tensor, optional*) –Encoder output. Defaults to None.
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回 The raw logit tensor. Shape (N, W, C) where C is `num_classes`.

返回类型 Tensor

class mmocr.models.textrecog.decoders.**MasterDecoder**(*n_layers=3*, *n_head=8*, *d_model=512*, *feat_size=240*, *d_inner=2048*, *attn_drop=0.0*, *ffn_drop=0.0*, *feat_pe_drop=0.2*, *module_loss=None*, *postprocessor=None*, *dictionary=None*, *max_seq_len=30*, *init_cfg=None*)

Decoder module in MASTER.

Code is partially modified from https://github.com/wenwenyu/MASTER-pytorch.

参数

- **n_layers** (*int*) –Number of attention layers. Defaults to 3.
- **n_head** (*int*) –Number of parallel attention heads. Defaults to 8.
- **d_model** (*int*) –Dimension E of the input from previous model. Defaults to 512.
- **feat_size** (*int*) –The size of the input feature from previous model, usually $H * W$. Defaults to 6 * 40.
- **d_inner** (*int*) –Hidden dimension of feedforward layers. Defaults to 2048.

- **attn_drop** (*float*) –Dropout rate of the attention layer. Defaults to 0.
- **ffn_drop** (*float*) –Dropout rate of the feedforward layer. Defaults to 0.
- **feat_pe_drop** (*float*) –Dropout rate of the feature positional encoding layer. Defaults to 0.2.
- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*. Defaults to None.
- **module_loss** (*dict, optional*) –Config to build module_loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **max_seq_len** (*int*) –Maximum output sequence length T. Defaults to 30.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

decode(*tgt_seq*, *feature*, *src_mask*, *tgt_mask*)

Decode the input sequence.

参数

- **tgt_seq** (*Tensor*) –Target sequence of shape: math: *(N, T, C)*.
- **feature** (*Tensor*) –Input feature map from encoder of shape: math: *(N, C, H, W)*
- **src_mask** (*BoolTensor*) –The source mask of shape: math: *(N, H*W)*.
- **tgt_mask** (*BoolTensor*) –The target mask of shape: math: *(N, T, T)*.

返回 The decoded sequence.

返回类型 Tensor

forward_test(*feat=None*, *out_enc=None*, *data_samples=None*)

Forward for testing.

参数

- **feat** (*Tensor, optional*) –Input feature map from backbone.
- **out_enc** (*Tensor*) –Unused.
- **data_samples** (*list[TextRecogDataSample]*) –Unused.

返回 Character probabilities. of shape $(N, self.max_seq_len, C)$ where C is `num_classes`.

返回类型 Tensor

forward_train(*feat=None*, *out_enc=None*, *data_samples=None*)

Forward for training. Source mask will not be used here.

参数

- **feat** (*Tensor, optional*) –Input feature map from backbone.

- **out_enc** (*Tensor*) –Unused.
- **data_samples** (*list[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing gt_text and valid_ratio information.

返回 The raw logit tensor. Shape (N, T, C) where C is `num_classes`.

返回类型 Tensor

make_target_mask(*tgt*, *device*)

Make target mask for self attention.

参数

- **tgt** (*Tensor*) –Shape [N, l_tgt]
- **device** (*torch.device*) –Mask device.

返回 Mask of shape [N * self.n_head, l_tgt, l_tgt]

返回类型 Tensor

class mmocr.models.textrecog.decoders.**NRTRDecoder**(*n_layers=6*, *d_embedding=512*, *n_head=8*, *d_k=64*, *d_v=64*, *d_model=512*, *d_inner=256*, *n_position=200*, *dropout=0.1*, *module_loss=None*, *postprocessor=None*, *dictionary=None*, *max_seq_len=30*, *init_cfg=None*)

Transformer Decoder block with self attention mechanism.

参数

- **n_layers** (*int*) –Number of attention layers. Defaults to 6.
- **d_embedding** (*int*) –Language embedding dimension. Defaults to 512.
- **n_head** (*int*) –Number of parallel attention heads. Defaults to 8.
- **d_k** (*int*) –Dimension of the key vector. Defaults to 64.
- **d_v** (*int*) –Dimension of the value vector. Defaults to 64
- **d_model** (*int*) –Dimension D_m of the input from previous model. Defaults to 512.
- **d_inner** (*int*) –Hidden dimension of feedforward layers. Defaults to 256.
- **n_position** (*int*) –Length of the positional encoding vector. Must be greater than `max_seq_len`. Defaults to 200.
- **dropout** (*float*) –Dropout rate for text embedding, MHSA, FFN. Defaults to 0.1.
- **module_loss** (*dict, optional*) –Config to build module_loss. Defaults to None.

- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **dictionary** (dict or Dictionary) –The config for *Dictionary* or the instance of *Dictionary*.
- **max_seq_len** (*int*) –Maximum output sequence length T. Defaults to 30.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

forward_test(*feat=None*, *out_enc=None*, *data_samples=None*)

Forward for testing.

参数

- **feat** (*Tensor, optional*) –Unused.
- **out_enc** (*Tensor*) –Encoder output of shape: math:*(N, T, D_m)* where D_m is d_model. Defaults to None.
- **data_samples** (*list[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing gt_text and valid_ratio information. Defaults to None.

返回 Character probabilities. of shape $(N, self.max_seq_len, C)$ where C is num_classes.

返回类型 Tensor

forward_train(*feat=None*, *out_enc=None*, *data_samples=None*)

Forward for training. Source mask will be used here.

参数

- **feat** (*Tensor, optional*) –Unused.
- **out_enc** (*Tensor*) –Encoder output of shape : math:*(N, T, D_m)* where D_m is d_model. Defaults to None.
- **data_samples** (*list[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing gt_text and valid_ratio information. Defaults to None.

返回 The raw logit tensor. Shape (N, T, C) where C is num_classes.

返回类型 Tensor

class mmocr.models.textrecog.decoders.**ParallelSARDecoder**(*dictionary*, *module_loss=None*, *postprocessor=None*, *enc_bi_rnn=False*, *dec_bi_rnn=False*, *dec_rnn_dropout=0.0*, *dec_gru=False*, *d_model=512*, *d_enc=512*, *d_k=64*, *pred_dropout=0.0*, *max_seq_len=30*, *mask=True*, *pred_concat=False*, *init_cfg=None*, ***kwargs*)

Implementation Parallel Decoder module in 'SAR.

<https://arxiv.org/abs/1811.00751>'_.

参数

- **dictionary** (dict or Dictionary) –The config for *Dictionary* or the instance of *Dictionary*.
- **module_loss** (*dict, optional*) –Config to build module_loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **enc_bi_rnn** (*bool*) –If True, use bidirectional RNN in encoder. Defaults to False.
- **dec_bi_rnn** (*bool*) –If True, use bidirectional RNN in decoder. Defaults to False.
- **dec_rnn_dropout** (*float*) –Dropout of RNN layer in decoder. Defaults to 0.0.
- **dec_gru** (*bool*) –If True, use GRU, else LSTM in decoder. Defaults to False.
- **d_model** (*int*) –Dim of channels from backbone D_i. Defaults to 512.
- **d_enc** (*int*) –Dim of encoder RNN layer D_m. Defaults to 512.
- **d_k** (*int*) –Dim of channels of attention module. Defaults to 64.
- **pred_dropout** (*float*) –Dropout probability of prediction layer. Defaults to 0.0.
- **max_seq_len** (*int*) –Maximum sequence length for decoding. Defaults to 30.
- **mask** (*bool*) –If True, mask padding in feature map. Defaults to True.
- **pred_concat** (*bool*) –If True, concat glimpse feature from attention with holistic feature and hidden state. Defaults to False.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

forward_test(*feat*, *out_enc*, *data_samples=None*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing valid_ratio information. Defaults to None.

返回 Character probabilities. of shape $(N, self.max_seql en, C)$ where C is `num_classes`.

返回类型 Tensor

forward_train(*feat*, *out_enc*, *data_samples*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **data_samples** (*list[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing gt_text and valid_ratio information.

返回 A raw logit tensor of shape (N, T, C).

返回类型 Tensor

class mmocr.models.textrecog.decoders.**ParallelSARDecoderWithBS**(*beam_width=5*, *num_classes=37*, *enc_bi_rnn=False*, *dec_bi_rnn=False*, *dec_do_rnn=0*, *dec_gru=False*, *d_model=512*, *d_enc=512*, *d_k=64*, *pred_dropout=0.0*, *max_seq_len=40*, *mask=True*, *start_idx=0*, *padding_idx=0*, *pred_concat=False*, *init_cfg=None*, ***kwargs*)

Parallel Decoder module with beam-search in SAR.

参数 **beam_width** (*int*) –Width for beam search.

forward_test (*feat*, *out_enc*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing valid_ratio information. Defaults to None.

返回 Character probabilities. of shape $(N, self.max_seq_len, C)$ where C is `num_classes`.

返回类型 Tensor

class mmocr.models.textrecog.decoders.**PositionAttentionDecoder**(*dictionary*, *module_loss=None*, *postprocessor=None*, *rnn_layers=2*, *dim_input=512*, *dim_model=128*, *max_seq_len=40*, *mask=True*, *return_feature=True*, *encode_value=False*, *init_cfg=None*)

Position attention decoder for RobustScanner.

RobustScanner: RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

参数

- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*.
- **module_loss** (*dict, optional*) –Config to build module_loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **rnn_layers** (*int*) –Number of RNN layers. Defaults to 2.
- **dim_input** (*int*) –Dimension D_i of input vector `feat`. Defaults to 512.
- **dim_model** (*int*) –Dimension D_m of the model. Should also be the same as encoder output vector `out_enc`. Defaults to 128.
- **max_seq_len** (*int*) –Maximum output sequence length T. Defaults to 40.
- **mask** (*bool*) –Whether to mask input features according to `img_meta['valid_ratio']`. Defaults to True.

- **return_feature** (*bool*) –Return feature or logits as the result. Defaults to True.
- **encode_value** (*bool*) –Whether to use the output of encoder `out_enc` as *value* of attention layer. If False, the original feature `feat` will be used. Defaults to False.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

forward_test(*feat*, *out_enc*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.
- **img_metas** (*Sequence[mmocr.structures.textrecog_data_sample.TextRecogDataSample]*) –

返回 Character probabilities of shape (N, T, C) if `return_feature=False`. Otherwise it would be the hidden feature before the prediction projection layer, whose shape is (N, T, D_m).

返回类型 Tensor

forward_train(*feat*, *out_enc*, *data_samples*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回 A raw logit tensor of shape (N, T, C) if `return_feature=False`. Otherwise it will be the hidden feature before the prediction projection layer, whose shape is (N, T, D_m).

返回类型 Tensor

class mmocr.models.textrecog.decoders.**RobustScannerFuser**(*dictionary*, *module_loss=None*, *postprocessor=None*, *hybrid_decoder={'type': 'SequenceAttentionDecoder'}*, *position_decoder={'type': 'PositionAttentionDecoder'}*, *max_seq_len=30*, *in_channels=[512, 512]*, *dim=-1*, *init_cfg=None*)

Decoder for RobustScanner.

参数

- **dictionary** (dict or Dictionary) –The config for *Dictionary* or the instance of *Dictionary*.
- **module_loss** (*dict, optional*) –Config to build module_loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **hybrid_decoder** (*dict*) –Config to build hybrid_decoder. Defaults to dict(type='SequenceAttentionDecoder').
- **position_decoder** (*dict*) –Config to build position_decoder. Defaults to dict(type='PositionAttentionDecoder').
- **fuser** (*dict*) –Config to build fuser. Defaults to dict(type='RobustScannerFuser').
- **max_seq_len** (*int*) –Maximum sequence length. The sequence is usually generated from decoder. Defaults to 30.
- **in_channels** (*list[int]*) –List of input channels. Defaults to [512, 512].
- **dim** (*int*) –The dimension on which to split the input. Defaults to -1.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

forward_test(*feat=None*, *out_enc=None*, *data_samples=None*)

Forward for testing.

参数

- **feat** (*torch.Tensor, optional*) –The feature map from backbone of shape (N, E, H, W). Defaults to None.
- **out_enc** (*torch.Tensor, optional*) –Encoder output. Defaults to None.

- **data_samples** (*Sequence[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing vaild_ratio information. Defaults to None.

返回 Character probabilities. of shape $(N, self.max_seq_len, C)$ where C is `num_classes`.

返回类型 Tensor

forward_train(*feat=None*, *out_enc=None*, *data_samples=None*)

Forward for training.

参数

- **feat** (*torch.Tensor, optional*) –The feature map from backbone of shape (N, E, H, W). Defaults to None.
- **out_enc** (*torch.Tensor, optional*) –Encoder output. Defaults to None.
- **data_samples** (*Sequence[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回类型 torch.Tensor

class mmocr.models.textrecog.decoders.**SequenceAttentionDecoder**(*dictionary*, *module_loss=None*, *postprocessor=None*, *rnn_layers=2*, *dim_input=512*, *dim_model=128*, *max_seq_len=40*, *mask=True*, *dropout=0*, *return_feature=True*, *encode_value=False*, *init_cfg=None*)

Sequence attention decoder for RobustScanner.

RobustScanner: RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

参数

- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*.
- **module_loss** (*dict, optional*) –Config to build module_loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **rnn_layers** (*int*) –Number of RNN layers. Defaults to 2.
- **dim_input** (*int*) –Dimension D_i of input vector `feat`. Defaults to 512.

- **dim_model** (*int*) –Dimension D_m of the model. Should also be the same as encoder output vector `out_enc`. Defaults to 128.
- **max_seq_len** (*int*) –Maximum output sequence length T. Defaults to 40.
- **mask** (*bool*) –Whether to mask input features according to `data_sample.valid_ratio`. Defaults to True.
- **dropout** (*float*) –Dropout rate for LSTM layer. Defaults to 0.
- **return_feature** (*bool*) –Return feature or logic as the result. Defaults to True.
- **encode_value** (*bool*) –Whether to use the output of encoder `out_enc` as *value* of attention layer. If False, the original feature `feat` will be used. Defaults to False.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

forward_test(*feat*, *out_enc*, *data_samples*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回 Character probabilities. of shape $(N, self.max_seq_len, C)$ where C is `num_classes`.

返回类型 Tensor

forward_test_step(*feat*, *out_enc*, *decode_sequence*, *current_step*, *data_samples*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **decode_sequence** (*Tensor*) –Shape (N, T). The tensor that stores history decoding result.
- **current_step** (*int*) –Current decoding step.
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回 Shape (N, C). The logit tensor of predicted tokens at current time step.

返回类型 Tensor

forward_train(*feat*, *out_enc*, *data_samples=None*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **targets_dict** (*dict*) –A dict with the key `padded_targets`, a tensor of shape (N, T). Each element is the index of a character.
- **data_samples** (*list[TextRecogDataSample], optional*) –Batch of TextRecogDataSample, containing gt_text information. Defaults to None.

返回 A raw logit tensor of shape (N, T, C) if `return_feature=False`. Otherwise it would be the hidden feature before the prediction projection layer, whose shape is (N, T, D_m).

返回类型 Tensor

class mmocr.models.textrecog.decoders.**SequentialSARDecoder**(*dictionary=None*, *module_loss=None*, *postprocessor=None*, *enc_bi_rnn=False*, *dec_bi_rnn=False*, *dec_gru=False*, *d_k=64*, *d_model=512*, *d_enc=512*, *pred_dropout=0.0*, *mask=True*, *max_seq_len=40*, *pred_concat=False*, *init_cfg=None*, ***kwargs*)

Implementation Sequential Decoder module in ‘SAR.

<https://arxiv.org/abs/1811.00751>‘_.

参数

- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*.
- **module_loss** (*dict, optional*) –Config to build module_loss. Defaults to None.
- **postprocessor** (*dict, optional*) –Config to build postprocessor. Defaults to None.
- **enc_bi_rnn** (*bool*) –If True, use bidirectional RNN in encoder. Defaults to False.
- **dec_bi_rnn** (*bool*) –If True, use bidirectional RNN in decoder. Defaults to False.

- **dec_do_rnn** (*float*) –Dropout of RNN layer in decoder. Defaults to 0.
- **dec_gru** (*bool*) –If True, use GRU, else LSTM in decoder. Defaults to False.
- **d_k** (*int*) –Dim of conv layers in attention module. Defaults to 64.
- **d_model** (*int*) –Dim of channels from backbone D_i. Defaults to 512.
- **d_enc** (*int*) –Dim of encoder RNN layer D_m. Defaults to 512.
- **pred_dropout** (*float*) –Dropout probability of prediction layer. Defaults to 0.
- **max_seq_len** (*int*) –Maximum sequence length during decoding. Defaults to 40.
- **mask** (*bool*) –If True, mask padding in feature map. Defaults to False.
- **pred_concat** (*bool*) –If True, concat glimpse feature from attention with holistic feature and hidden state. Defaults to False.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

forward_test(*feat*, *out_enc*, *data_samples=None*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **data_samples** (*list[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing valid_ratio information.

返回 Character probabilities. of shape $(N, self.max_seq_len, C)$ where C is `num_classes`.

返回类型 Tensor

forward_train(*feat*, *out_enc*, *data_samples=None*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **data_samples** (*list[TextRecogDataSample]*) –Batch of TextRecogDataSample, containing gt_text and valid_ratio information.

返回 A raw logit tensor of shape (N, T, C).

返回类型 Tensor

40.15 Text Recognition Module Losses

class mmocr.models.textrecog.module_losses.**ABIModuleLoss**(*dictionary*, *max_seq_len=40*, *letter_case='unchanged'*, *weight_vis=1.0*, *weight_lang=1.0*, *weight_fusion=1.0*, ***kwargs*)

Implementation of ABINet multiloss that allows mixing different types of losses with weights.

参数

- **dictionary** (dict or Dictionary) –The config for *Dictionary* or the instance of *Dictionary*.
- **max_seq_len** (*int*) –Maximum sequence length. The sequence is usually generated from decoder. Defaults to 40.
- **letter_case** (*str*) –There are three options to alter the letter cases of gt texts: - unchanged: Do not change gt texts. - upper: Convert gt texts into uppercase characters. - lower: Convert gt texts into lowercase characters. Usually, it only works for English characters. Defaults to 'unchanged'.
- **weight_vis** (*float or int*) –The weight of vision decoder loss. Defaults to 1.0.
- **weight_dec** (*float or int*) –The weight of language decoder loss. Defaults to 1.0.
- **weight_fusion** (*float or int*) –The weight of fuser (aligner) loss. Defaults to 1.0.
- **weight_lang** (*Union[float, int]*) –

返回类型 None

forward(*outputs*, *data_samples*)

参数

- **outputs** (*dict*) –The output dictionary with at least one of out_vis, out_langs and out_fusers specified.
- **data_samples** (*list[TextRecogDataSample]*) –List of TextRecogDataSample which are processed by get_target.

返回 A loss dictionary with loss_visual, loss_lang and loss_fusion. Each should either be the loss tensor or None if the output of its corresponding module is not given.

返回类型 dict

class mmocr.models.textrecog.module_losses.**BaseTextRecogModuleLoss**(*dictionary*, *max_seq_len=40*, *letter_case='unchanged'*, *pad_with='auto'*, ***kwargs*)

Base recognition loss.

参数

- **dictionary** (dict or Dictionary) –The config for *Dictionary* or the instance of *Dictionary*.
- **max_seq_len** (*int*) –Maximum sequence length. The sequence is usually generated from decoder. Defaults to 40.
- **letter_case** (*str*) –There are three options to alter the letter cases of gt texts: - unchanged: Do not change gt texts. - upper: Convert gt texts into uppercase characters. - lower: Convert gt texts into lowercase characters. Usually, it only works for English characters. Defaults to ‘unchanged’.
- **pad_with** (*str*) –The padding strategy for gt_text.padded_indexes. Defaults to ‘auto’. Options are: - ‘auto’: Use dictionary.padding_idx to pad gt texts, or

  dictionary.end_idx if dictionary.padding_idx is None.

  - ’padding’: Always use dictionary.padding_idx to pad gt texts.
  - ’end’: Always use dictionary.end_idx to pad gt texts.
  - ’none’: Do not pad gt texts.

返回类型 None

get_targets(*data_samples*)

Target generator.

参数 **data_samples** (*list[TextRecogDataSample]*) –It usually includes gt_text information.

返回

Updated data_samples. Two keys will be added to data_sample:

- indexes (torch.LongTensor): Character indexes representing gt texts. All special tokens are excluded, except for UKN.
- padded_indexes (torch.LongTensor): Character indexes representing gt texts with BOS and EOS if applicable, following several padding indexes until the length reaches max_seq_len. In particular, if pad_with='none', no padding will be applied.

返回类型 list[*TextRecogDataSample*]

class mmocr.models.textrecog.module_losses.**CEModuleLoss**(*dictionary*, *max_seq_len=40*, *letter_case='unchanged'*, *pad_with='auto'*, *ignore_char='padding'*, *flatten=False*, *reduction='none'*, *ignore_first_char=False*)

Implementation of loss module for encoder-decoder based text recognition method with CrossEntropy loss.

参数

- **dictionary** (dict or Dictionary) –The config for *Dictionary* or the instance of *Dictionary*.
- **max_seq_len** (*int*) –Maximum sequence length. The sequence is usually generated from decoder. Defaults to 40.
- **letter_case** (*str*) –There are three options to alter the letter cases of gt texts: - unchanged: Do not change gt texts. - upper: Convert gt texts into uppercase characters. - lower: Convert gt texts into lowercase characters. Usually, it only works for English characters. Defaults to 'unchanged'.
- **pad_with** (*str*) –The padding strategy for gt_text.padded_indexes. Defaults to 'auto'. Options are: - 'auto': Use dictionary.padding_idx to pad gt texts, or

  dictionary.end_idx if dictionary.padding_idx is None.

  - 'padding': Always use dictionary.padding_idx to pad gt texts.
  - 'end': Always use dictionary.end_idx to pad gt texts.
  - 'none': Do not pad gt texts.

- **ignore_char** (*int or str*) –Specifies a target value that is ignored and does not contribute to the input gradient. ignore_char can be int or str. If int, it is the index of the ignored char. If str, it is the character to ignore. Apart from single characters, each item can be one of the following reversed keywords: 'padding', 'start', 'end', and 'unknown', which refer to their corresponding special tokens in the dictionary. It will not ignore any special tokens when ignore_char == -1 or 'none'. Defaults to 'padding'.
- **flatten** (*bool*) –Whether to flatten the output and target before computing CE loss. Defaults to False.
- **reduction** (*str*) –Specifies the reduction to apply to the output, should be one of the following: ('none', 'mean', 'sum'). Defaults to 'none'.
- **ignore_first_char** (*bool*) –Whether to ignore the first token in target (usually the start token). If True, the last token of the output sequence will also be removed to be

aligned with the target length. Defaults to `False`.

- **flatten** –Whether to flatten the vectors for loss computation. Defaults to False.

forward(*outputs*, *data_samples*)

参数

- **outputs** (*Tensor*) –A raw logit tensor of shape (N, T, C).
- **data_samples** (*list[TextRecogDataSample]*) –List of `TextRecogDataSample` which are processed by `get_target`.

返回 A loss dict with the key `loss_ce`.

返回类型 dict

class mmocr.models.textrecog.module_losses.**CTCModuleLoss**(*dictionary*, *letter_case='unchanged'*, *flatten=True*, *reduction='mean'*, *zero_infinity=False*, ***kwargs*)

Implementation of loss module for CTC-loss based text recognition.

参数

- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*.
- **letter_case** (*str*) –There are three options to alter the letter cases of gt texts: - unchanged: Do not change gt texts. - upper: Convert gt texts into uppercase characters. - lower: Convert gt texts into lowercase characters. Usually, it only works for English characters. Defaults to 'unchanged'.
- **flatten** (*bool*) –If True, use flattened targets, else padded targets.
- **reduction** (*str*) –Specifies the reduction to apply to the output, should be one of the following: ('none', 'mean', 'sum').
- **zero_infinity** (*bool*) –Whether to zero infinite losses and the associated gradients. Default: False. Infinite losses mainly occur when the inputs are too short to be aligned to the targets.

返回类型 None

forward(*outputs*, *data_samples*)

参数

- **outputs** (*Tensor*) –A raw logit tensor of shape (N, T, C).

- **data_samples** (*list[TextRecogDataSample]*) –List of TextRecogDataSample which are processed by get_target.

返回 The loss dict with key loss_ctc.

返回类型 dict

get_targets(*data_samples*)

Target generator.

参数 **data_samples** (*list[TextRecogDataSample]*) –It usually includes gt_text information.

返回

updated data_samples. It will add two key in data_sample:

- indexes (torch.LongTensor): The index corresponding to the item.

返回类型 list[*TextRecogDataSample*]

40.16 KIE Extractors

class mmocr.models.kie.extractors.**SDMGR**(*backbone=None*, *roi_extractor=None*, *neck=None*, *kie_head=None*, *dictionary=None*, *data_preprocessor=None*, *init_cfg=None*)

The implementation of the paper: Spatial Dual-Modality Graph Reasoning for Key Information Extraction. https://arxiv.org/abs/2103.14470.

参数

- **backbone** (*dict, optional*) –Config of backbone. If None, None will be passed to kie_head during training and testing. Defaults to None.
- **roi_extractor** (*dict, optional*) –Config of roi extractor. Only applicable when backbone is not None. Defaults to None.
- **neck** (*dict, optional*) –Config of neck. Defaults to None.
- **kie_head** (*dict*) –Config of KIE head. Defaults to None.
- **dictionary** (*dict, optional*) –Config of dictionary. Defaults to None.
- **data_preprocessor** (*dict or ConfigDict, optional*) –The pre-process config of BaseDataPreprocessor. it usually includes, pad_size_divisor, pad_value, mean and std. It has to be None when working in non-visual mode. Defaults to None.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs. Defaults to None.

返回类型 None

extract_feat(*img*, *gt_bboxes*)

Extract features from images if self.backbone is not None. It returns None otherwise.

参数

- **img** (*torch.Tensor*) –The input image with shape (N, C, H, W).
- **gt_bboxes** (*list[torch.Tensor)*) –A list of ground truth bounding boxes, each of shape $(N_i, 4)$.

返回 The extracted features with shape (N, E).

返回类型 torch.Tensor

forward(*inputs*, *data_samples=None*, *mode='tensor'*, ***kwargs*)

The unified entry for a forward process in both training and test.

The method should accept three modes: "tensor", "predict" and "loss":

- "tensor": Forward the whole network and return tensor or tuple of

tensor without any post-processing, same as a common nn.Module. - "predict": Forward and return the predictions, which are fully processed to a list of `DetDataSample`. - "loss": Forward and return a dict of losses according to the given inputs and data samples.

Note that this method doesn't handle neither back propagation nor optimizer updating, which are done in the `train_step()`.

参数

- **inputs** (*torch.Tensor*) –The input tensor with shape (N, C, ···) in general.
- **data_samples** (list[`DetDataSample`], optional) –The annotation data of every samples. Defaults to None.
- **mode** (*str*) –Return what kind of value. Defaults to 'tensor'.

返回

The return type depends on `mode`.

- If `mode="tensor"`, return a tensor or a tuple of tensor.
- If `mode="predict"`, return a list of `DetDataSample`.
- If `mode="loss"`, return a dict of tensor.

返回类型 torch.Tensor

loss(*inputs*, *data_samples*, ***kwargs*)

Calculate losses from a batch of inputs and data samples.

参数

- **inputs** (*torch.Tensor*) –Input images of shape (N, C, H, W). Typically these should be mean centered and std scaled.
- **data_samples** (*list[KIEDataSample]*) –A list of N datasamples, containing meta information and gold annotations for each of the images.

返回 A dictionary of loss components.

返回类型 dict[str, Tensor]

predict(*inputs*, *data_samples*, ***kwargs*)

Predict results from a batch of inputs and data samples with post- processing. :param inputs: Input images of shape (N, C, H, W).

Typically these should be mean centered and std scaled.

参数

- **data_samples** (*list[KIEDataSample]*) –A list of N datasamples, containing meta information and gold annotations for each of the images.
- **inputs** (*torch.Tensor*) –

返回 A list of datasamples of prediction results. Results are stored in `pred_instances.labels` and `pred_instances.edge_labels`.

返回类型 List[*KIEDataSample*]

40.17 KIE Heads

class mmocr.models.kie.heads.**SDMGRHead**(*dictionary*, *num_classes=26*, *visual_dim=64*, *fusion_dim=1024*, *node_input=32*, *node_embed=256*, *edge_input=5*, *edge_embed=256*, *num_gnn=2*, *bidirectional=False*, *relation_norm=10.0*, *module_loss={'type': 'SDMGRModuleLoss'}*, *postprocessor={'type': 'SDMGRPostProcessor'}*, *init_cfg={'mean': 0, 'override': {'name': 'edge_embed'}, 'std': 0.01, 'type': 'Normal'}*)

SDMGR Head.

参数

- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*.
- **num_classes** (*int*) –Number of class labels. Defaults to 26.
- **visual_dim** (*int*) –Dimension of visual features E. Defaults to 64.

- **fusion_dim** (*int*) –Dimension of fusion layer. Defaults to 1024.
- **node_input** (*int*) –Dimension of raw node embedding. Defaults to 32.
- **node_embed** (*int*) –Dimension of node embedding. Defaults to 256.
- **edge_input** (*int*) –Dimension of raw edge embedding. Defaults to 5.
- **edge_embed** (*int*) –Dimension of edge embedding. Defaults to 256.
- **num_gnn** (*int*) –Number of GNN layers. Defaults to 2.
- **bidirectional** (*bool*) –Whether to use bidirectional RNN to embed nodes. Defaults to False.
- **relation_norm** (*float*) –Norm to map value from one range to another.= Defaults to 10.
- **module_loss** (*dict*) –Module Loss config. Defaults to `dict(type='SDMGRModuleLoss')`.
- **postprocessor** (*dict*) –Postprocessor config. Defaults to `dict(type='SDMGRPostProcessor')`.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

返回类型 None

compute_relations(*data_samples*)

Compute the relations between every two boxes for each datasample, then return the concatenated relations.

参数 **data_samples** (*List[mmocr.structures.kie_data_sample.KIEDataSample]*) –

返回类型 torch.Tensor

convert_texts(*data_samples*)

Extract texts in datasamples and pack them into a batch.

参数 **data_samples** (*List[KIEDataSample]*) –List of data samples.

返回

- node_nums (List[int]): A list of node numbers for each sample.
- char_nums (List[Tensor]): A list of character numbers for each sample.
- nodes (Tensor): A tensor of shape (N, C) where C is the maximum number of characters in a sample.

返回类型 tuple(List[int], List[Tensor], Tensor)

forward(*inputs*, *data_samples*)

参数

- **inputs** (*torch.Tensor*) –Shape (N, E).
- **data_samples** (*List[KIEDataSample]*) –List of data samples.

返回

- node_cls (Tensor): Raw logits scores for nodes. Shape (N, C_l) where C_l is number of classes.
- edge_cls (Tensor): Raw logits scores for edges. Shape $(N * N, 2)$.

返回类型 tuple(Tensor, Tensor)

loss(*inputs*, *data_samples*)

Calculate losses from a batch of inputs and data samples. :param inputs: Shape (N, E). :type inputs: torch.Tensor :param data_samples: List of data samples. :type data_samples: List[KIEDataSample]

返回 A dictionary of loss components.

返回类型 dict[str, tensor]

参数

- **inputs** (*torch.Tensor*) –
- **data_samples** (*List[mmocr.structures.kie_data_sample.KIEDataSample]*) –

predict(*inputs*, *data_samples*)

Predict results from a batch of inputs and data samples with post- processing.

参数

- **inputs** (*torch.Tensor*) –Shape (N, E).
- **data_samples** (*List[KIEDataSample]*) –List of data samples.

返回

A list of datasamples of prediction results. Results are stored in `pred_instances.labels`, `pred_instances.scores`, `pred_instances.edge_labels` and `pred_instances.edge_scores`.

- labels (Tensor): An integer tensor of shape (N,) indicating bbox labels for each image.
- scores (Tensor): A float tensor of shape (N,), indicating the confidence scores for node label predictions.
- edge_labels (Tensor): An integer tensor of shape (N, N) indicating the connection between nodes. Options are 0, 1.
- edge_scores (Tensor): A float tensor of shape (N,), indicating the confidence scores for edge predictions.

返回类型 List[*KIEDataSample*]

40.18 KIE Module Losses

class mmocr.models.kie.module_losses.**SDMGRModuleLoss**(*weight_node=1.0*, *weight_edge=1.0*, *ignore_idx=- 100*)

The implementation the loss of key information extraction proposed in the paper: Spatial Dual-Modality Graph Reasoning for Key Information Extraction.

参数

- **weight_node** (*float*) –Weight of node loss. Defaults to 1.0.
- **weight_edge** (*float*) –Weight of edge loss. Defaults to 1.0.
- **ignore_idx** (*int*) –Node label to ignore. Defaults to -100.

返回类型 None

forward(*preds*, *data_samples*)

Forward function.

参数

- **preds** (*tuple(Tensor, Tensor)*) –
- **data_samples** (*list[KIEDataSample]*) –A list of datasamples containing `gt_instances.labels` and `gt_instances.edge_labels`.

返回 Loss dict, containing `loss_node`, `loss_edge`, `acc_node` and `acc_edge`.

返回类型 dict(str, Tensor)

CHAPTER 41

mmocr.structures

41.1 Text Detection Data Sample

class mmocr.structures.textdet_data_sample.**TextDetDataSample**(*, *metainfo=None*, ***kwargs*)

A data structure interface of MMOCR. They are used as interfaces between different components.

The attributes in `TextDetDataSample` are divided into two parts:

- "gt_instances"(InstanceData): Ground truth of instance annotations.
- "pred_instances"(InstanceData): Instances of model predictions.

实际案例

```
>>> import torch
>>> import numpy as np
>>> from mmengine.structures import InstanceData
>>> from mmocr.data import TextDetDataSample
>>> # gt_instances
>>> data_sample = TextDetDataSample()
>>> img_meta = dict(img_shape=(800, 1196, 3),
...                 pad_shape=(800, 1216, 3))
>>> gt_instances = InstanceData(metainfo=img_meta)
>>> gt_instances.bboxes = torch.rand((5, 4))
```

(下页继续)

(续上页)

```
>>> gt_instances.labels = torch.rand((5,))
>>> data_sample.gt_instances = gt_instances
>>> assert 'img_shape' in data_sample.gt_instances.metainfo_keys()
>>> len(data_sample.gt_instances)
5
>>> print(data_sample)
```

<TextDetDataSample(META INFORMATION DATA FIELDS gt_instances: <InstanceData(

META INFORMATION pad_shape: (800, 1216, 3) img_shape: (800, 1196, 3) DATA FIELDS labels: tensor([0.8533, 0.1550, 0.5433, 0.7294, 0.5098]) bboxes: tensor([[9.7725e-01, 5.8417e-01, 1.7269e-01, 6.5694e-01],

[1.7894e-01, 5.1780e-01, 7.0590e-01, 4.8589e-01], [7.0392e-01, 6.6770e-01, 1.7520e-01, 1.4267e-01], [2.2411e-01, 5.1962e-01, 9.6953e-01, 6.6994e-01], [4.1338e-01, 2.1165e-01, 2.7239e-04, 6.8477e-01]])

) at 0x7f21fb1b9190>

) at 0x7f21fb1b9880>

```
>>> # pred_instances
>>> pred_instances = InstanceData(metainfo=img_meta)
>>> pred_instances.bboxes = torch.rand((5, 4))
>>> pred_instances.scores = torch.rand((5,))
>>> data_sample = TextDetDataSample(pred_instances=pred_instances)
>>> assert 'pred_instances' in data_sample
>>> data_sample = TextDetDataSample()
>>> gt_instances_data = dict(
...                         bboxes=torch.rand(2, 4),
...                         labels=torch.rand(2),
...                         masks=np.random.rand(2, 2, 2))
>>> gt_instances = InstanceData(**gt_instances_data)
>>> data_sample.gt_instances = gt_instances
>>> assert 'gt_instances' in data_sample
>>> assert 'masks' in data_sample.gt_instances
```

参数 **metainfo** (*Optional[dict]*) –

返回类型 None

property gt_instances: mmengine.structures.instance_data.InstanceData

groundtruth instances.

Type InstanceData

property pred_instances: mmengine.structures.instance_data.InstanceData

prediction instances.

Type InstanceData

41.2 Text Recognition Data Sample

class mmocr.structures.textrecog_data_sample.**TextRecogDataSample**(*, *metainfo=None*, ***kwargs*)

A data structure interface of MMOCR for text recognition. They are used as interfaces between different components.

The attributes in `TextRecogDataSample` are divided into two parts:

- "gt_text"(LabelData): Ground truth text.
- "pred_text"(LabelData): predictions text.

实际案例

```
>>> import torch
>>> import numpy as np
>>> from mmengine.structures import LabelData
>>> from mmocr.data import TextRecogDataSample
>>> # gt_text
>>> data_sample = TextRecogDataSample()
>>> img_meta = dict(img_shape=(800, 1196, 3),
...                 pad_shape=(800, 1216, 3))
>>> gt_text = LabelData(metainfo=img_meta)
>>> gt_text.item = 'mmocr'
>>> data_sample.gt_text = gt_text
>>> assert 'img_shape' in data_sample.gt_text.metainfo_keys()
>>> print(data_sample)
```

<TextRecogDataSample(META INFORMATION DATA FIELDS gt_text: <LabelData(

META INFORMATION pad_shape: (800, 1216, 3) img_shape: (800, 1196, 3) DATA FIELDS item: 'mmocr'

) at 0x7f21fb1b9190>

) at 0x7f21fb1b9880>

```
>>> # pred_text
>>> pred_text = LabelData(metainfo=img_meta)
```

(下页继续)

(续上页)

```
>>> pred_text.item = 'mmocr'
>>> data_sample = TextRecogDataSample(pred_text=pred_text)
>>> assert 'pred_text' in data_sample
>>> data_sample = TextRecogDataSample()
>>> gt_text_data = dict(item='mmocr')
>>> gt_text = LabelData(**gt_text_data)
>>> data_sample.gt_text = gt_text
>>> assert 'gt_text' in data_sample
>>> assert 'item' in data_sample.gt_text
```

参数 **metainfo** (*Optional[dict]*) –

返回类型 None

property gt_text: mmengine.structures.label_data.LabelData

ground truth text.

Type LabelData

property pred_text: mmengine.structures.label_data.LabelData

prediction text.

Type LabelData

41.3 KIE Data Sample

class mmocr.structures.kie_data_sample.**KIEDataSample**(*, *metainfo=None*, ***kwargs*)

A data structure interface of MMOCR. They are used as interfaces between different components.

The attributes in `KIEDataSample` are divided into two parts:

- "gt_instances"(InstanceData): Ground truth of instance annotations.
- "pred_instances"(InstanceData): Instances of model predictions.

实际案例

```
>>> import torch
>>> import numpy as np
>>> from mmengine.structures import InstanceData
>>> from mmocr.data import KIEDataSample
>>> # gt_instances
>>> data_sample = KIEDataSample()
>>> img_meta = dict(img_shape=(800, 1196, 3),
```

(下页继续)

(续上页)

```
...                 pad_shape=(800, 1216, 3))
>>> gt_instances = InstanceData(metainfo=img_meta)
>>> gt_instances.bboxes = torch.rand((5, 4))
>>> gt_instances.labels = torch.rand((5,))
>>> data_sample.gt_instances = gt_instances
>>> assert 'img_shape' in data_sample.gt_instances.metainfo_keys()
>>> len(data_sample.gt_instances)
5
>>> print(data_sample)
```

<KIEDataSample(META INFORMATION DATA FIELDS gt_instances: <InstanceData(

META INFORMATION pad_shape: (800, 1216, 3) img_shape: (800, 1196, 3) DATA FIELDS labels: tensor([0.8533, 0.1550, 0.5433, 0.7294, 0.5098]) bboxes: tensor([[9.7725e-01, 5.8417e-01, 1.7269e-01, 6.5694e-01],

[1.7894e-01, 5.1780e-01, 7.0590e-01, 4.8589e-01], [7.0392e-01, 6.6770e-01, 1.7520e-01, 1.4267e-01], [2.2411e-01, 5.1962e-01, 9.6953e-01, 6.6994e-01], [4.1338e-01, 2.1165e-01, 2.7239e-04, 6.8477e-01]])

) at 0x7f21fb1b9190>

) at 0x7f21fb1b9880>

```
>>> # pred_instances
>>> pred_instances = InstanceData(metainfo=img_meta)
>>> pred_instances.bboxes = torch.rand((5, 4))
>>> pred_instances.scores = torch.rand((5,))
>>> data_sample = KIEDataSample(pred_instances=pred_instances)
>>> assert 'pred_instances' in data_sample
>>> data_sample = KIEDataSample()
>>> gt_instances_data = dict(
...                         bboxes=torch.rand(2, 4),
...                         labels=torch.rand(2))
>>> gt_instances = InstanceData(**gt_instances_data)
>>> data_sample.gt_instances = gt_instances
>>> assert 'gt_instances' in data_sample
```

参数 **metainfo** (*Optional[dict]*) –

返回类型 None

property gt_instances: mmengine.structures.instance_data.InstanceData

groundtruth instances.

Type InstanceData

property pred_instances: mmengine.structures.instance_data.InstanceData

prediction instances.

Type InstanceData

CHAPTER 42

mmocr.visualization

42.1 Text Detection Visualizer

class mmocr.visualization.textdet_visualizer.**TextDetLocalVisualizer**(*name='visualizer'*, *image=None*, *with_poly=True*, *with_bbox=False*, *vis_backends=None*, *save_dir=None*, *gt_color='g'*, *pred_color='r'*, *line_width=2*, *alpha=0.8*)

The MMOCR Text Detection Local Visualizer.

参数

- **name** (*str*) –Name of the instance. Defaults to 'visualizer'.
- **image** (*np.ndarray, optional*) –The origin image to draw. The format should be RGB. Defaults to None.
- **with_poly** (*bool*) –Whether to draw polygons. Defaults to True.
- **with_bbox** (*bool*) –Whether to draw bboxes. Defaults to False.
- **vis_backends** (*list, optional*) –Visual backend config list. Defaults to None.

- **save_dir** (*str, optional*) –Save file dir for all storage backends. If it is None, the backend storage will not save any data.
- **gt_color** (*Union[str, tuple, list[str], list[tuple]]*) –The colors of GT polygons and bboxes. `colors` can have the same length with lines or just single value. If `colors` is single value, all the lines will have the same colors. Refer to *matplotlib.colors* for full list of formats that are accepted. Defaults to 'g'.
- **pred_color** (*Union[str, tuple, list[str], list[tuple]]*) –The colors of pred polygons and bboxes. `colors` can have the same length with lines or just single value. If `colors` is single value, all the lines will have the same colors. Refer to *matplotlib.colors* for full list of formats that are accepted. Defaults to 'r'.
- **line_width** (*int, float*) –The linewidth of lines. Defaults to 2.
- **alpha** (*float*) –The transparency of bboxes or polygons. Defaults to 0.8.

返回类型 None

add_datasample(*name*, *image*, *data_sample=None*, *draw_gt=True*, *draw_pred=True*, *show=False*, *wait_time=0*, *out_file=None*, *pred_score_thr=0.3*, *step=0*)

Draw datasample and save to all backends.

- If GT and prediction are plotted at the same time, they are

displayed in a stitched image where the left image is the ground truth and the right image is the prediction. - If `show` is True, all storage backends are ignored, and the images will be displayed in a local window. - If `out_file` is specified, the drawn image will be saved to `out_file`. This is usually used when the display is not available.

参数

- **name** (*str*) –The image identifier.
- **image** (*np.ndarray*) –The image to draw.
- **data_sample** (`TextDetDataSample`, optional) –

  TextDetDataSample which contains gt and prediction. Defaults to None.
- **draw_gt** (*bool*) –Whether to draw GT TextDetDataSample. Defaults to True.
- **draw_pred** (*bool*) –Whether to draw Predicted TextDetDataSample. Defaults to True.
- **show** (*bool*) –Whether to display the drawn image. Default to False.
- **wait_time** (*float*) –The interval of show (s). Defaults to 0.
- **out_file** (*str*) –Path to output file. Defaults to None.
- **pred_score_thr** (*float*) –The threshold to visualize the bboxes and masks. Defaults to 0.3.

- **step** (*int*) –Global step value to record. Defaults to 0.

返回类型 None

42.2 Text Recognition Visualizer

class mmocr.visualization.textrecog_visualizer.**TextRecogLocalVisualizer**(*name='visualizer'*, *image=None*, *vis_backends=None*, *save_dir=None*, *gt_color='g'*, *pred_color='r'*)

MMOCR Text Detection Local Visualizer.

参数

- **name** (*str*) –Name of the instance. Defaults to 'visualizer'.
- **image** (*np.ndarray, optional*) –The origin image to draw. The format should be RGB. Defaults to None.
- **vis_backends** (*list, optional*) –Visual backend config list. Defaults to None.
- **save_dir** (*str, optional*) –Save file dir for all storage backends. If it is None, the backend storage will not save any data.
- **gt_color** (*str or tuple[int, int, int]*) –Colors of GT text. The tuple of color should be in RGB order. Or using an abbreviation of color, such as '*g*' for '*green*'. Defaults to 'g'.
- **pred_color** (*str or tuple[int, int, int]*) –Colors of Predicted text. The tuple of color should be in RGB order. Or using an abbreviation of color, such as '*r*' for '*red*'. Defaults to 'r'.

返回类型 None

add_datasample(*name*, *image*, *data_sample=None*, *draw_gt=True*, *draw_pred=True*, *show=False*, *wait_time=0*, *pred_score_thr=None*, *out_file=None*, *step=0*)

Visualize datasample and save to all backends.

- If GT and prediction are plotted at the same time, they are

displayed in a stitched image where the left image is the ground truth and the right image is the prediction. - If `show` is True, all storage backends are ignored, and the images will be displayed in a local window. - If `out_file` is specified, the drawn image will be saved to `out_file`. This is usually used when the display is not available.

参数

- **name** (*str*) –The image title. Defaults to ‘image’.
- **image** (*np.ndarray*) –The image to draw.
- **data_sample** (TextRecogDataSample, optional) –TextRecogDataSample which contains gt and prediction. Defaults to None.
- **draw_gt** (*bool*) –Whether to draw GT TextRecogDataSample. Defaults to True.
- **draw_pred** (*bool*) –Whether to draw Predicted TextRecogDataSample. Defaults to True.
- **show** (*bool*) –Whether to display the drawn image. Defaults to False.
- **wait_time** (*float*) –The interval of show (s). Defaults to 0.
- **out_file** (*str*) –Path to output file. Defaults to None.
- **step** (*int*) –Global step value to record. Defaults to 0.
- **pred_score_thr** (*float*) –Threshold of prediction score. It’s not used in this function. Defaults to None.

返回类型 None

42.3 Text Spotting Visualizer

class mmocr.visualization.textspotting_visualizer.**TextSpottingLocalVisualizer**(*name='visualizer'*, *image=None*, *vis_backends=None*, *save_dir=None*, *fig_save_cfg={'frameon': False}*, *fig_show_cfg={'frameon': False}*)

参数

- **image** (*Optional[numpy.ndarray]*) –
- **vis_backends** (*Optional[List[Dict]]*) –
- **save_dir** (*Optional[str]*) –

返回类型 None

add_datasample(*name*, *image*, *data_sample=None*, *draw_gt=True*, *draw_pred=True*, *show=False*, *wait_time=0*, *pred_score_thr=None*, *out_file=None*, *step=0*)

Draw datasample.

参数

- **name** (*str*) –
- **image** (*numpy.ndarray*) –
- **data_sample** (*Optional[mmocr.structures.textdet_data_sample.TextDetDataSample]*) –
- **draw_gt** (*bool*) –
- **draw_pred** (*bool*) –
- **show** (*bool*) –
- **wait_time** (*int*) –
- **pred_score_thr** (*Optional[float]*) –
- **out_file** (*Optional[str]*) –
- **step** (*int*) –

返回类型 None

42.4 KIE Visualizer

class mmocr.visualization.kie_visualizer.**KIELocalVisualizer**(*name='kie_visualizer'*, *is_openset=False*, ***kwargs*)

The MMOCR Text Detection Local Visualizer.

参数

- **name** (*str*) –Name of the instance. Defaults to 'visualizer'.
- **image** (*np.ndarray, optional*) –the origin image to draw. The format should be RGB. Defaults to None.
- **vis_backends** (*list, optional*) –Visual backend config list. Default to None.
- **save_dir** (*str, optional*) –Save file dir for all storage backends. If it is None, the backend storage will not save any data.
- **fig_save_cfg** (*dict*) –Keyword parameters of figure for saving. Defaults to empty dict.
- **fig_show_cfg** (*dict*) –Keyword parameters of figure for showing. Defaults to empty dict.
- **is_openset** (*bool, optional*) –Whether the visualizer is used in OpenSet. Defaults to False.

返回类型 None

add_datasample(*name*, *image*, *data_sample=None*, *draw_gt=True*, *draw_pred=True*, *show=False*, *wait_time=0*, *pred_score_thr=None*, *out_file=None*, *step=0*)

Draw datasample and save to all backends.

- If GT and prediction are plotted at the same time, they are

displayed in a stitched image where the left image is the ground truth and the right image is the prediction. - If `show` is True, all storage backends are ignored, and the images will be displayed in a local window. - If `out_file` is specified, the drawn image will be saved to `out_file`. This is usually used when the display is not available.

参数

- **name** (*str*) –The image identifier.
- **image** (*np.ndarray*) –The image to draw.
- **data_sample** (`KIEDataSample`, optional) –

  KIEDataSample which contains gt and prediction. Defaults to None.
- **draw_gt** (*bool*) –Whether to draw GT KIEDataSample. Defaults to True.
- **draw_pred** (*bool*) –Whether to draw Predicted KIEDataSample. Defaults to True.
- **show** (*bool*) –Whether to display the drawn image. Default to False.
- **wait_time** (*float*) –The interval of show (s). Defaults to 0.
- **pred_score_thr** (*float*) –The threshold to visualize the bboxes and masks. Defaults to 0.3.
- **out_file** (*str*) –Path to output file. Defaults to None.
- **step** (*int*) –Global step value to record. Defaults to 0.

返回类型 None

draw_arrows(*x_data*, *y_data*, *colors='C1'*, *line_widths=1*, *line_styles='-'*, *arrow_tail_widths=0.001*, *arrow_head_widths=None*, *arrow_head_lengths=None*, *arrow_shapes='full'*, *overhangs=0*)

Draw single or multiple arrows.

参数

- **x_data** (*np.ndarray or torch.Tensor*) –The x coordinate of each line' start and end points.
- **y_data** (*np.ndarray, torch.Tensor*) –The y coordinate of each line' start and end points.
- **colors** (*str or tuple or list[str or tuple]*) –The colors of lines. `colors` can have the same length with lines or just single value. If `colors` is single

value, all the lines will have the same colors. Reference to https://matplotlib.org/stable/gallery/color/named_colors.html for more details. Defaults to 'g'.

- **line_widths** (*int or float or list[int or float]*) –The linewidth of lines. `line_widths` can have the same length with lines or just single value. If `line_widths` is single value, all the lines will have the same linewidth. Defaults to 2.
- **line_styles** (*str or list[str]*) –The linestyle of lines. `line_styles` can have the same length with lines or just single value. If `line_styles` is single value, all the lines will have the same linestyle. Defaults to '- '.
- **arrow_tail_widths** (*int or float or list[int, float]*) –The width of arrow tails. `arrow_tail_widths` can have the same length with lines or just single value. If `arrow_tail_widths` is single value, all the lines will have the same width. Defaults to 0.001.
- **arrow_head_widths** (*int or float or list[int, float]*) –The width of arrow heads. `arrow_head_widths` can have the same length with lines or just single value. If `arrow_head_widths` is single value, all the lines will have the same width. Defaults to None.
- **arrow_head_lengths** (*int or float or list[int, float]*) –The length of arrow heads. `arrow_head_lengths` can have the same length with lines or just single value. If `arrow_head_lengths` is single value, all the lines will have the same length. Defaults to None.
- **arrow_shapes** (*str or list[str]*) –The shapes of arrow heads. `arrow_shapes` can have the same length with lines or just single value. If `arrow_shapes` is single value, all the lines will have the same shape. Defaults to 'full'.
- **overhangs** (*int or list[int]*) –The overhangs of arrow heads. `overhangs` can have the same length with lines or just single value. If `overhangs` is single value, all the lines will have the same overhangs. Defaults to 0.

返回类型 mmengine.visualization.visualizer.Visualizer

CHAPTER 43

English

CHAPTER 44

简体中文

CHAPTER 45

导引

- genindex
- search

Python 模块索引

m

索引

A

B

C

D

E

F

H

I

K

L

M

N

O

P

R

S

T

U

V

W