MMOCR

发布 *1.0.1*

OpenMMLab

2023 年 09 月 01 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【快速运行】来熟悉 MMOCR 的基本用法，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。
- 请阅读*FAQ* 来查找常见问题的答案。
- 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。
- 我们也欢迎每一位用户成为贡献者！请阅读贡献指南 来了解如何为 MMOCR 做出贡献。

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。

GPU 平台

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

CPU 平台

```
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，MMCV 和 MMDetection。

```
pip install -U openmim
mim install mmengine
mim install mmcv
mim install mmdet
```

第二步安装 MMOCR.

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（推荐）。

如果你将 MMOCR 作为一个外置依赖库使用，则可以通过 MIM 安装。

源码安装

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

MIM 安装

```
mim install mmocr
```

第三步（可选）如果你需要使用与 albumentations 有关的变换（如 ABINet 数据流水线中的 Albu），或需要构建文档、运行单元测试的依赖，请使用以下命令安装依赖：

源码安装

```
# 安装 albu
pip install -r requirements/albu.txt
# 安装文档、测试等依赖
pip install -r requirements.txt
```

MIM 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

你可以通过运行一个简单的推理任务来检验 MMOCR 的安装是否成功。

Python

在 Python 中运行以下代码：

```
>>> from mmocr.apis import MMOCRInferencer
>>> ocr = MMOCRInferencer(det='DBNet', rec='CRNN')
>>> ocr('demo/demo_text_ocr.jpg', show=True, print_result=True)
```

Shell

如果你是通过源码安装的 MMOCR，你可以在 MMOCR 的根目录下运行以下命令：

```
python tools/infer.py demo/demo_text_ocr.jpg --det DBNet --rec CRNN --show --print-
↪result
```

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：

```
# 识别结果
{'predictions': [{'rec_texts': ['cbanks', 'docecea', 'grouf', 'pwate', 'chobnsonsg',
↪'soxee', 'oeioh', 'c', 'sones', 'lbrandec', 'sretalg', '11', 'to8', 'round', 'sale',
↪ 'year',
'ally', 'sie', 'sall'], 'rec_scores': [...], 'det_polygons': [...], 'det_scores':
[...]}]}
```

注解： 如果你在没有 GUI 的服务器上运行 MMOCR，或者通过没有开启 X11 转发的 SSH 隧道运行 MMOCR，你可能无法看到弹出的窗口。

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 MMEngine、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 MMEngine、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

这个章节会介绍 MMOCR 的一些基本功能。我们假设你已经从源码安装了 MMOCR。此外，你也可以通过教程 Notebook来了解如何在交互式环境下实现推理、训练和测试。

3.1 推理

在 MMOCR 的根目录下运行以下命令：

```
python tools/infer.py demo/demo_text_ocr.jpg --det DBNet --rec CRNN --show --print-
↪result
```

你可以看到弹出的预测结果，以及在控制台中打印出的推理结果。

```
# 识别结果
{'predictions': [{'rec_texts': ['cbanks', 'docecea', 'grouf', 'pwate', 'chobnsonsg',
↪'soxee', 'oeioh', 'c', 'sones', 'lbrandec', 'sretalg', '11', 'to8', 'round', 'sale',
↪ 'year',
'ally', 'sie', 'sall'], 'rec_scores': [...], 'det_polygons': [...], 'det_scores':
[...]}]}
```

注解： 如果你在没有 GUI 的服务器上运行 MMOCR，或者通过没有开启 X11 转发的 SSH 隧道运行 MMOCR，你可能无法看到弹出的窗口。

对 MMOCR 中推理接口更为详细的说明，可以在这里找到。

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。

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 数据集提供了一键式数据准备脚本。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/` 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/
tar xzvf mini_icdar2015.tar.gz -C data/
```

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/textdet/_base_/datasets/icdar2015.py`。打开该文件，把第一行 `icdar2015_textdet_data_root` 指向的路径替换：

```
icdar2015_textdet_data_root = 'data/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，且只保留最后一个权重
default_hooks = dict(
    checkpoint=dict(
```

(下页继续)

(续上页)

```
        type='CheckpointHook',
        interval=10,
        max_keep_ckpts=1,
    ))
# 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 (MMEngine: Config) 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/textdet/_base_/schedules/schedule_sgd_1200e.py` 和 `configs/textdet/_base_/default_runtime.py` 中，感兴趣的读者可以自行查看。

注解： 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。

注解： 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（你可能会得到一个不太一样的结果）：

```
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/textdet/_base_/datasets/icdar2015.py，修改 icdar2015_textdet_test 的 data_root 为 tests/data/det_toy_dataset:

```
# ...
icdar2015_textdet_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
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。

注解： 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_resnet18_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。

注解： 有关更多可视化功能的介绍，请参阅这里。

CHAPTER 4

FAQ

4.1 General

Q1 I'm getting the warning like `unexpected key in source state_dict: fc.weight, fc.bias`, is there something wrong?

A It's not an error. It occurs because the backbone network is pretrained on image classification tasks, where the last fc layer is required to generate the classification output. However, the fc layer is no longer needed when the backbone network is used to extract features in downstream tasks, and therefore these weights can be safely skipped when loading the checkpoint.

Q2 MMOCR terminates with an error: `shapely.errors.TopologicalError: The operation 'GEOSIntersection_r' could not be performed. Likely cause is invalidity of the geometry`. How could I fix it?

A This error occurs because of some invalid polygons (e.g., polygons with self-intersections) existing in the dataset or generated by some non-rigorous data transforms. These polygons can be fixed by adding `FixInvalidPolygon` transform after the transform likely to introduce invalid polygons. For example, a common practice is to append it after `LoadOCRAnnotations` in both train and test pipeline. The resulting pipeline should look like:

```
train_pipeline = [
    ...
    dict(
        type='LoadOCRAnnotations',
        with_polygon=True,
```

(下页继续)

(续上页)

```
        with_bbox=True,
        with_label=True,
    ),
    dict(type='FixInvalidPolygon', min_poly_points=4),
    ...
]
```

In practice, we find that Totaltext contains some invalid polygons and using `FixInvalidPolygon` is a must. Here is an example config.

Q3 Getting `libpng warning: iCCP: known incorrect sRGB profile` when loading images with `cv2` backend.

A This is a warning from `libpng` and it is safe to ignore. It is caused by the `icc` profile in the image. You can use `pillow` backend to avoid this warning:

```
train_pipeline = [
    dict(
        type='LoadImageFromFile',
        imdecode_backend='pillow'),
    ...
]
```

4.2 Text Recognition

Q1 What are the steps to train text recognition models with my own dictionary?

A In MMOCR 1.0, you only need to modify the config and point `Dictionary` to your custom dict file. For example, if you want to train SAR model (https://github.com/open-mmlab/mmocr/blob/75c06d34bbc01d3d11dfd7afc098b6cdeee82579/configs/textrecog/sar/sar_resnet31_parallel-decoder_5e_st-sub_mj-sub_sa_real.py) with your own dictionary placed at `/my/dict.txt`, you can modify `dictionary.dict_file` term in base config to:

```
dictionary = dict(
    type='Dictionary',
    dict_file='/my/dict.txt',
    with_start=True,
    with_end=True,
    same_start_end=True,
    with_padding=True,
    with_unknown=True)
```

Now you are good to go. You can also find more information in Dictionary API.

Q2 How to properly visualize non-English characters?

A You can customize `font_families` or `font_properties` in visualizer. For example, to visualize Korean:

`configs/textrecog/_base_/default_runtime.py`:

```
visualizer = dict(
    type='TextRecogLocalVisualizer',
    name='visualizer',
    font_families='NanumGothic', # new feature
    vis_backends=vis_backends)
```

It's also fine to pass the font path to visualizer:

```
visualizer = dict(
    type='TextRecogLocalVisualizer',
    name='visualizer',
    font_properties='path/to/font_file',
    vis_backends=vis_backends)
```

CHAPTER 5

推理

在 OpenMMLab 中，所有的推理操作都被统一到了推理器 `Inferencer` 中。推理器被设计成为一个简洁易用的 API，它在不同的 OpenMMLab 库中都有着非常相似的接口。

MMOCR 中存在两种不同的推理器：

- **标准推理器**：MMOCR 中的每个基本任务都有一个标准推理器，即 `TextDetInferencer`（文本检测），`TextRecInferencer`（文本识别），`TextSpottingInferencer`（端到端 OCR）和 `KIEInferencer`（关键信息提取）。它们具有非常相似的接口，具有标准的输入/输出协议，并且总体遵循 OpenMMLab 的设计。这些推理器也可以被串联在一起，以便对一系列任务进行推理。
- **MMOCRInferencer**：我们还提供了 `MMOCRInferencer`，一个专门为 MMOCR 设计的便捷推理接口。它封装和链接了 MMOCR 中的所有推理器，因此用户可以使用此推理器对图像执行一系列任务，并直接获得最终结果。但是，它的接口与标准推理器有一些不同，并且为了简单起见，可能会牺牲一些标准的推理器功能。

对于新用户，我们建议使用 **MMOCRInferencer** 来测试不同模型的组合。

如果你是开发人员并希望将模型集成到自己的项目中，我们建议使用**标准推理器**，因为它们更灵活且标准化，并具有完整的功能。

5.1 基础用法

MMOCRInferencer

目前，MMOCRInferencer 可以对以下任务进行推理：

- 文本检测
- 文本识别
- OCR（文本检测 + 文本识别）
- 关键信息提取（文本检测 + 文本识别 + 关键信息提取）
- *OCR*（*text spotting*）（即将推出）

为了便于使用，MMOCRInferencer 向用户提供了 Python 接口和命令行接口。例如，如果你想要对 demo/demo_text_ocr.jpg 进行 OCR 推理，使用 DBNet 作为文本检测模型，CRNN 作为文本识别模型，只需执行以下命令：

Python

```
>>> from mmocr.apis import MMOCRInferencer
>>> # 读取模型
>>> ocr = MMOCRInferencer(det='DBNet', rec='SAR')
>>> # 进行推理并可视化结果
>>> ocr('demo/demo_text_ocr.jpg', show=True)
```

命令行

```
python tools/infer.py demo/demo_text_ocr.jpg --det DBNet --rec SAR --show
```

可视化结果将被显示在一个新窗口中：

注解： 如果你在没有 GUI 的服务器上运行 MMOCR，或者是通过禁用 X11 转发的 SSH 隧道运行该指令，show 选项将不起作用。然而，你仍然可以通过设置 out_dir 和 save_vis=True 参数将可视化数据保存到文件。阅读储存结果 了解详情。

根据初始化参数，MMOCRInferencer 可以在不同模式下运行。例如，如果初始化时指定了 det、rec 和 kie，它可以在 KIE 模式下运行。

Python

```
>>> kie = MMOCRInferencer(det='DBNet', rec='SAR', kie='SDMGR')
>>> kie('demo/demo_kie.jpeg', show=True)
```

命令行

```
python tools/infer.py demo/demo_kie.jpeg --det DBNet --rec SAR --kie SDMGR --show
```

可视化结果如下：

可以见到，MMOCRInferencer 的 Python 接口与命令行接口的使用方法非常相似。下文将以 Python 接口为例，介绍 MMOCRInferencer 的具体用法。关于命令行接口的更多信息，请参考命令行接口。

标准推理器

通常，OpenMMLab 中的所有标准推理器都具有非常相似的接口。下面的例子展示了如何使用 `TextDetInferencer` 对单个图像进行推理。

```
>>> from mmocr.apis import TextDetInferencer
>>> # 读取模型
>>> inferencer = TextDetInferencer(model='DBNet')
>>> # 推理
>>> inferencer('demo/demo_text_ocr.jpg', show=True)
```

可视化结果如图：

5.2 初始化

每个推理器必须使用一个模型进行初始化。初始化时，可以手动选择推理设备。

5.2.1 模型初始化

MMOCRInferencer

对于每个任务，`MMOCRInferencer` 需要两个参数 `xxx` 和 `xxx_weights`（例如 `det` 和 `det_weights`）以对模型进行初始化。此处将以 `det` 和 `det_weights` 为例来说明一些典型的初始化模型的方法。

- 要用 MMOCR 的预训练模型进行推理，只需要把它的名字传给参数 `det`，权重将自动从 OpenMMLab 的模型库中下载和加载。此处记录了 MMOCR 中可以通过该方法初始化的所有模型。

  ```
  >>> MMOCRInferencer(det='DBNet')
  ```

- 要加载自定义的配置和权重，你可以把配置文件的路径传给 `det`，把权重的路径传给 `det_weights`。

  ```
  >>> MMOCRInferencer(det='path/to/dbnet_config.py', det_weights='path/to/dbnet.pth
  ↪')
  ```

如果需要查看更多的初始化方法，请点击“标准推理器”选项卡。

标准推理器

每个标准的 Inferencer 都接受两个参数，model 和 weights 。在 MMOCRInferencer 中，这两个参数分别对应 xxx 和 xxx_weights （例如 det 和 det_weights）。

- model 接受模型的名称或配置文件的路径作为输入。模型的名称从 model-index.yml 中的模型的元文件（示例 ）中获取。你可以在此处找到可用权重的列表。
- weights 接受权重文件的路径。

此处列举了一些常见的初始化模型的方法。

- 你可以通过传递模型的名称给 model 来推理 MMOCR 的预训练模型。权重将会自动从 OpenMMLab 的模型库中下载并加载。

```
>>> from mmocr.apis import TextDetInferencer
>>> inferencer = TextDetInferencer(model='DBNet')
```

注解： 模型与推理器的任务种类必须匹配。

你可以通过将权重的路径或 URL 传递给 weights 来让推理器加载自定义的权重。

```
>>> inferencer = TextDetInferencer(model='DBNet', weights='path/to/dbnet.pth')
```

- 如果有自定义的配置和权重，你可以将配置文件的路径传递给 model，将权重的路径传递给 weights。

```
>>> inferencer = TextDetInferencer(model='path/to/dbnet_config.py', weights='path/
↪to/dbnet.pth')
```

- 默认情况下，MMEngine 会在训练模型时自动将配置文件转储到权重文件中。如果你有一个在 MMEngine 上训练的权重，你也可以将权重文件的路径传递给 weights，而不需要指定 model：

```
>>> # 如果无法在权重中找到配置文件，则会引发错误
>>> inferencer = TextDetInferencer(weights='path/to/dbnet.pth')
```

- 传递配置文件到 model 而不指定 weight 则会产生一个随机初始化的模型。

5.2.2 推理设备

每个推理器实例都会跟一个设备绑定。默认情况下，最佳设备是由 MMEngine 自动决定的。你也可以通过指定 device 参数来改变设备。例如，你可以使用以下代码在 GPU 1 上创建一个推理器。

MMOCRInferencer

```
>>> inferencer = MMOCRInferencer(det='DBNet', device='cuda:1')
```

标准推理器

```
>>> inferencer = TextDetInferencer(model='DBNet', device='cuda:1')
```

如要在 CPU 上创建一个推理器：

MMOCRInferencer

```
>>> inferencer = MMOCRInferencer(det='DBNet', device='cpu')
```

标准推理器

```
>>> inferencer = TextDetInferencer(model='DBNet', device='cpu')
```

请参考 torch.device 了解 `device` 参数支持的所有形式。

5.3 推理

当推理器初始化后，你可以直接传入要推理的原始数据，从返回值中获取推理结果。

5.3.1 输入

MMOCRInferencer / TextDetInferencer / TextRecInferencer / TextSpottingInferencer

输入可以是以下任意一种格式：

- str: 图像的路径/URL。

```
>>> inferencer('demo/demo_text_ocr.jpg')
```

- array: 图像的 numpy 数组。它应该是 BGR 格式。

```
>>> import mmcv
>>> array = mmcv.imread('demo/demo_text_ocr.jpg')
>>> inferencer(array)
```

- list: 基本类型的列表。列表中的每个元素都将单独处理。

```
>>> inferencer(['img_1.jpg', 'img_2.jpg])
>>> # 列表内混合类型也是允许的
>>> inferencer(['img_1.jpg', array])
```

- str: 目录的路径。目录中的所有图像都将被处理。

```
>>> inferencer('tests/data/det_toy_dataset/imgs/test/')
```

KIEInferencer

输入可以是一个字典或者一个字典列表，其中每个字典包含以下键：

- `img` (str 或者 ndarray): 图像的路径或图像本身。如果 KIE 推理器在无可视模式下使用，则不需要此键。如果它是一个 numpy 数组，则应该是 BGR 顺序编码的图片。
- `img_shape` (tuple(int, int)): 图像的形状 (H, W)。仅在 KIE 推理器在无可视模式下使用且没有提供 `img` 时才需要。
- `instances` (list[dict]): 实例列表。

每个 `instance` 都应该包含以下键：

```
{
    # 一个嵌套列表，其中包含 4 个数字，表示实例的边界框，顺序为 (x1, y1, x2, y2)
    "bbox": np.array([[x1, y1, x2, y2], [x1, y1, x2, y2], ...],
                     dtype=np.int32),

    # 文本列表
    "texts": ['text1', 'text2', ...],
}
```

5.3.2 输出

默认情况下，每个推理器都以字典格式返回预测结果。

- `visualization` 包含可视化的预测结果。但默认情况下，它是一个空列表，除非 `return_vis=True`。
- `predictions` 包含以 json-可序列化格式返回的预测结果。如下所示，内容因任务类型而异。

  MMOCRInferencer

```
{
    'predictions' : [
      # 每个实例都对应于一个输入图像
      {
        'det_polygons': [...],  # 2d 列表，长度为 (N,)，格式为 [x1, y1, x2, y2, ...]
        'det_scores': [...],  # 浮点列表，长度为 (N, )
        'det_bboxes': [...],   # 2d 列表，形状为 (N, 4)，格式为 [min_x, min_y, max_x,
↪ max_y]
        'rec_texts': [...],  # 字符串列表，长度为 (N, )
        'rec_scores': [...],  # 浮点列表，长度为 (N, )
        'kie_labels': [...],  # 节点标签，长度为 (N, )
        'kie_scores': [...],  # 节点置信度，长度为 (N, )
        'kie_edge_scores': [...],  # 边预测置信度，形状为 (N, N)
        'kie_edge_labels': [...]  # 边标签，形状为 (N, N)
```

(下页继续)

(续上页)

```
      },
      ...
    ],
    'visualization' : [
      array(..., dtype=uint8),
    ]
}
```

标准推理器

TextDetInferencer

```
{
    'predictions' : [
      # 每个实例都对应于一个输入图像
      {
        'polygons': [...],  # 2d 列表，长度为 (N,)，格式为 [x1, y1, x2, y2, ...]
        'bboxes': [...],  # 2d 列表，形状为 (N, 4)，格式为 [min_x, min_y, max_x, max_
↪y]
        'scores': [...]  # 浮点列表，长度为 (N, )
      },
      ...
    ]
    'visualization' : [
      array(..., dtype=uint8),
    ]
}
```

TextRecInferencer

```
{
    'predictions' : [
      # 每个实例都对应于一个输入图像
      {
        'text': '...',  # 字符串
        'scores': 0.1,  # 浮点
      },
      ...
    ]
    'visualization' : [
      array(..., dtype=uint8),
    ]
}
```

TextSpottingInferencer

```
{
    'predictions' : [
      # 每个实例都对应于一个输入图像
      {
        'polygons': [...],  # 2d 列表，长度为 (N,)，格式为 [x1, y1, x2, y2, ...]
        'bboxes': [...],  # 2d 列表，形状为 (N, 4)，格式为 [min_x, min_y, max_x, max_
↪y]
        'scores': [...]  # 浮点列表，长度为（N, ）
        'texts': ['...',]  # 字符串列表，长度为（N, ）
      },
    ]
    'visualization' : [
      array(..., dtype=uint8),
    ]
}
```

KIEInferencer

```
{
    'predictions' : [
      # 每个实例都对应于一个输入图像
      {
        'labels': [...],  # 节点标签，长度为 (N, )
        'scores': [...],  # 节点置信度，长度为 (N, )
        'edge_scores': [...],  # 边预测置信度，形状为 (N, N)
        'edge_labels': [...],  # 边标签，形状为 (N, N)
      },
    ]
    'visualization' : [
      array(..., dtype=uint8),
    ]
}
```

如果你想要从模型中获取原始输出，可以将 `return_datasamples` 设置为 `True` 来获取原始的 DataSample，它将存储在 `predictions` 中。

5.3.3 储存结果

除了从返回值中获取预测结果，你还可以通过设置 `out_dir` 和 `save_pred/save_vis` 参数将预测结果和可视化结果导出到文件中。

```
>>> inferencer('img_1.jpg', out_dir='outputs/', save_pred=True, save_vis=True)
```

结果目录结构如下：

```
outputs
├── preds
│   └── img_1.json
└── vis
    └── img_1.jpg
```

文件名与对应的输入图像文件名相同。如果输入图像是数组，则文件名将是从 0 开始的数字。

5.3.4 批量推理

你可以通过设置 `batch_size` 来自定义批量推理的批大小。默认批大小为 1。

5.4 API

这里列出了推理器详尽的参数列表。

MMOCRInferencer

MMOCRInferencer.__init__():

[1]: 当同时指定了文本检测和识别模型时，`kie` 才会生效。

MMOCRInferencer.__call__()

标准推理器

Inferencer.__init__():

Inferencer.__call__()

5.5 命令行接口

注解： 该节仅适用于 `MMOCRInferencer`.

`MMOCRInferencer` 的命令行形式可以通过 `tools/infer.py` 调用，大致形式如下：

```
python tools/infer.py INPUT_PATH [--det DET] [--det-weights ...] ...
```

其中，`INPUT_PATH` 为必须字段，内容应当为指向图片或文件目录的路径。其他参数与 Python 接口遵循的映射关系如下：

- 在命令行中调用参数时，需要在 Python 接口的参数前面加上两个`-`，然后把下划线 `_` 替换成连字符`-`。例如，`out_dir` 会变成 `--out-dir`。

- 对于布尔类型的参数，将参数放在命令中就相当于将其指定为 True。例如，`--show` 会将 `show` 参数指定为 True。

此外，命令行中默认不会回显推理结果，你可以通过 `--print-result` 参数来查看推理结果。

下面是一个例子：

```
python tools/infer.py demo/demo_text_ocr.jpg --det DBNet --rec SAR --show --print-
↪result
```

运行该命令，可以得到如下结果：

```
{'predictions': [{'rec_texts': ['CBank', 'Docbcba', 'GROUP', 'MAUN', 'CROBINSONS',
↪'AOCOC', '916M3', 'BOO9', 'Oven', 'BRANDS', 'ARETAIL', '14', '70<UKN>S', 'ROUND',
↪'SALE', 'YEAR', 'ALLY', 'SALE', 'SALE'],
'rec_scores': [0.9753464579582214, ...], 'det_polygons': [[551.9930285844646, 411.
↪9138765335083, 553.6153911653112,
383.53195309638977, 620.2410061195247, 387.33785033226013, 618.6186435386782, 415.
↪71977376937866], ...], 'det_scores': [0.8230461478233337, ...]}]}
```

CHAPTER 6

配置文件

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。

6.1 常见用法

注解： 本小节建议结合 MMEngine: 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
icdar2015_textdet_train = _base_.icdar2015_textdet_train
icdar2015_textdet_train.pipeline = _base_.train_pipeline
icdar2015_textdet_test = _base_.icdar2015_textdet_test
icdar2015_textdet_test.pipeline = _base_.test_pipeline

train_dataloader = dict(
    batch_size=16,
    num_workers=8,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=True),
    dataset=icdar2015_textdet_train)

val_dataloader = dict(
    batch_size=1,
    num_workers=4,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=False),
    dataset=icdar2015_textdet_test)
```

6.1.1 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。

6.1.2 _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

```
icdar2015_textdet_train = _base_.icdar2015_textdet_train
# ...
train_dataloader = dict(
    # ...
    dataset=icdar2015_textdet_train)
```

6.1.3 _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
   icdar2015_textdet_train = _base_.icdar2015_textdet_train
   # 可以直接利用 Python 的 update 修改变量
   icdar2015_textdet_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
   icdar2015_textdet_train = _base_.icdar2015_textdet_train
   # 类属性方法修改
   icdar2015_textdet_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
```

6.1.4 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*MMEngine: 命令行修改配置*.

6.2 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。

6.2.1 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'))
randomness = 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 MMEngine: 注册器。
- env_cfg 设置分布式环境配置，更多配置可以详见 MMEngine: Runner。
- randomness 设置 numpy，torch，cudnn 等随机种子，更多配置详见 MMEngine: Runner。

6.2.2 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'), # 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'), # 多卡情况下，同步 buffer
    visualization=dict( # 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 MMEngine: 钩子 (Hook)。

6.2.3 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 Python: 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 MMEngine: 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 值为各种损失的平均值。

6.2.4 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 MMEngine: 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 MMEngine: 循环控制器。

6.2.5 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/验证} = 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/验证：数据集用于训练，测试还是验证

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
mjsynth_textrecog_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)

icdar2013_textrecog_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)

icdar2015_textrecog_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）。更多信息可以参考数据流水线

- 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)。

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
train_pipeline = [
    dict(
        type='LoadImageFromFile',
        color_type='grayscale',
        ignore_empty=True,
        min_size=5),
    dict(type='LoadOCRAnnotations', with_text=True),
    dict(type='Resize', scale=(100, 32), keep_ratio=False),
    dict(
        type='PackTextRecogInputs',
        meta_keys=('img_path', 'ori_shape', 'img_shape', 'valid_ratio'))
```

(下页继续)

(续上页)

```
]
test_pipeline = [
    dict(
        type='LoadImageFromFile',
        color_type='grayscale'),
    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=[mjsynth_textrecog_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(
        type='ConcatDataset',
        datasets=[icdar2013_textrecog_test, icdar2015_textrecog_test],
        pipeline=test_pipeline))
test_dataloader = val_dataloader
```

6.2.6 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。更多信息可以参考网络结构

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'),
        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,
```

(下页继续)

(续上页)

```
        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(
            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 MMEngine: 加载权重或恢复训练与*OCR 进阶技巧-断点恢复训练*。

6.2.7 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MMEngine: 评测指标 (Metric) 和评测器 (Evaluator) 和评测器

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 MMEngine: 评测指标，在 MMOCR 中不同算法任务有不同的评测指标。更多 OCR 相关的评测指标可以参考评测指标。

文字检测: *HmeanIOUMetric*

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，对多个数据集 (IC13 和 IC15) 用多个 `Metric` (`WordMetric` 和 `CharMetric`) 进行评测：

```
# 评测部分
val_evaluator = dict(
    type='MultiDatasetsEvaluator',
    metrics=[
        dict(
            type='WordMetric',
            mode=['exact', 'ignore_case', 'ignore_case_symbol']),
        dict(type='CharMetric')
    ],
    dataset_prefixes=['IC13', 'IC15'])
test_evaluator = val_evaluator
```

6.2.8 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')
```

6.3 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)、textrecog (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 *base* 配置汇总。同时可能会修改一些 _base_ 中的变量，如 batch size, 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：

```
configs
├── textdet
|   ├── _base_
|   |   ├── datasets
|   |   |   ├── icdar2015.py
|   |   |   ├── icdar2017.py
|   |   |   └── totaltext.py
|   |   ├── schedules
```

(下页继续)

(续上页)

```
|   |   |   └── 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
```

6.4 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 encoder 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 7

数据集准备

7.1 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。因此，为了方便用户进行数据集准备，我们提供了一键式的数据准备脚本，使得用户仅需使用一行命令即可完成数据集准备的全部步骤。

在这一节，我们将介绍一个典型的数据集准备流程：

1. 下载数据集并将其格式转换为 *MMOCR* 支持的格式
2. 修改配置文件

然而，如果你已经有了 MMOCR 支持的格式的数据集，那么第一步就不是必须的。你可以阅读数据集类及标注格式来了解更多细节。

7.2 数据集下载及格式转换

以 ICDAR 2015 数据集的文本检测任务准备步骤为例，你可以执行以下命令来完成数据集准备：

```
python tools/dataset_converters/prepare_dataset.py icdar2015 --task textdet
```

命令执行完成后，数据集将被下载并转换至 MMOCR 格式，文件目录结构如下：

```
data/icdar2015
├── textdet_imgs
│   ├── test
│   └── train
├── textdet_test.json
└── textdet_train.json
```

数据准备完毕以后，你也可以通过使用我们提供的数据集浏览工具*browse_dataset.py* 来可视化数据集的标签是否被正确生成，例如：

```
python tools/analysis_tools/browse_dataset.py configs/textdet/_base_/datasets/
↪icdar2015.py
```

7.3 修改配置文件

7.3.1 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（当你使用 `prepare_dataset.py` 来准备数据集时，这个配置文件通常会在数据集准备就绪后自动生成），这里我们以 ICDAR 2015 数据集为例（见 `configs/textdet/_base_/datasets/icdar2015.py`）：

```
icdar2015_textdet_data_root = 'data/icdar2015' # 数据集根目录

# 训练集配置
icdar2015_textdet_train = dict(
    type='OCRDataset',
    data_root=icdar2015_textdet_data_root,                # 数据根目录
    ann_file='textdet_train.json',                        # 标注文件名称
    filter_cfg=dict(filter_empty_gt=True, min_size=32),   # 数据过滤
    pipeline=None)
# 测试集配置
icdar2015_textdet_test = dict(
    type='OCRDataset',
    data_root=icdar2015_textdet_data_root,
    ann_file='textdet_test.json',
    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',
]

icdar2015_textdet_train = _base_.icdar2015_textdet_train              # 指定训练集
icdar2015_textdet_train.pipeline = _base_.train_pipeline    # 指定训练集使用的数据流水线
icdar2015_textdet_test = _base_.icdar2015_textdet_test                # 指定测试集
icdar2015_textdet_test.pipeline = _base_.test_pipeline      # 指定测试集使用的数据流水线

train_dataloader = dict(
    batch_size=16,
    num_workers=8,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=True),
    dataset=icdar2015_textdet_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=icdar2015_textdet_test)    # 在 val_dataloader 中指定使用的验证数据集

test_dataloader = val_dataloader
```

7.3.2 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',
```

(下页继续)

(续上页)

```
    '../_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_.mjsynth_textrecog_train]
# 测试集列表
test_list = [
    _base_.cute80_textrecog_test, _base_.iiit5k_textrecog_test, _base_.svt_textrecog_
↪test,
    _base_.svtp_textrecog_test, _base_.icdar2013_textrecog_test, _base_.icdar2015_
↪textrecog_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 8

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。

8.1 单卡机器训练及测试

8.1.1 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。

8.1.2 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。

8.2 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。

8.2.1 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
```

8.2.2 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。

8.2.3 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。

8.3 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
```

8.4 进阶技巧

8.4.1 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'
```

8.4.2 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：

8.4.3 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
```

8.4.4 可视化模型测试结果

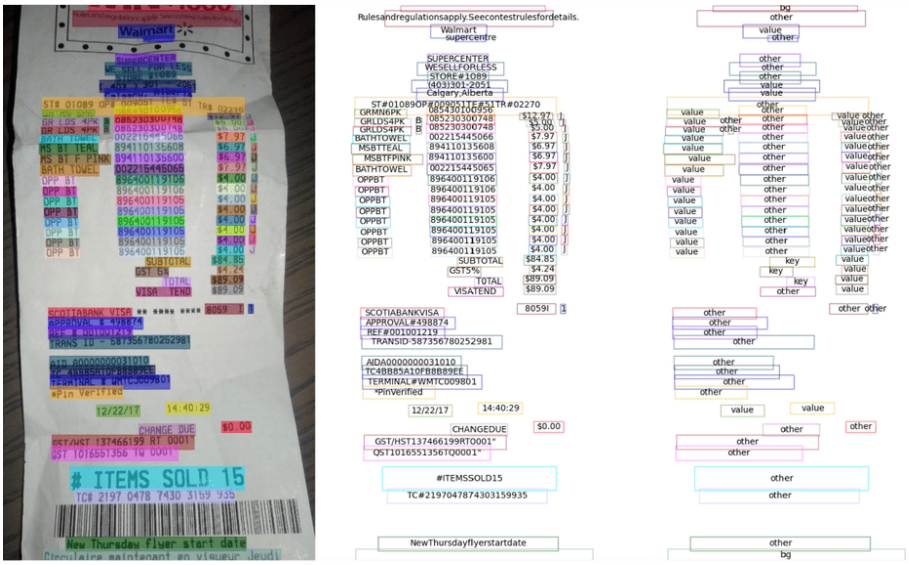
`tools/test.py` 提供了可视化接口，以方便用户对模型进行定性分析。

(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：

8.4.5 测试时数据增强

测试时增强，指的是在推理（预测）阶段，将原始图片进行水平翻转、垂直翻转、对角线翻转、旋转角度等数据增强操作，得到多张图，分别进行推理，再对多个结果进行综合分析，得到最终输出结果。为此，MMOCR 提供了一键式测试时数据增强，仅需在测试时添加 `--tta` 参数即可。

注解： TTA 仅支持文本识别模型。

```
python tools/test.py configs/textrecog/crnn/crnn_mini-vgg_5e_mj.py checkpoints/crnn_
↪mini-vgg_5e_mj.pth --tta
```

CHAPTER 9

可视化

阅读本文前建议先阅读 MMEngine: 可视化 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。

9.1 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。

9.2 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'),]
```

9.3 绘制

9.3.1 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 10

常用工具

10.1 可视化工具

10.1.1 数据集可视化工具

MMOCR 提供了数据集可视化工具 `tools/visualizations/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（通常存放在如 `configs/textdet/dbnet/xxx.py` 文件中）或数据集配置（通常存放在 `configs/textdet/_base_/datasets/xxx.py` 文件中）路径。该工具将依据输入的配置文件类型自动将经过数据流水线（data pipeline）处理过的图像及其对应的标签，或原始图片及其对应的标签绘制出来。

支持参数

```
python tools/visualizations/browse_dataset.py \
    ${CONFIG_FILE} \
    [-o, --output-dir ${OUTPUT_DIR}] \
    [-p, --phase ${DATASET_PHASE}] \
    [-m, --mode ${DISPLAY_MODE}] \
    [-t, --task ${DATASET_TASK}] \
    [-n, --show-number ${NUMBER_IMAGES_DISPLAY}] \
    [-i, --show-interval ${SHOW_INTERRVAL}] \
    [--cfg-options ${CFG_OPTIONS}]
```

用法示例

以下示例演示了如何使用该工具可视化“DBNet_R50_icdar2015”模型使用的训练数据。

```
# 使用默认参数可视化 "dbnet_r50dcn_v2_fpnc_1200e_icadr2015" 模型的训练数据
python tools/visualizations/browse_dataset.py configs/textdet/dbnet/dbnet_resnet50-
↪dcnv2_fpnc_1200e_icdar2015.py
```

默认情况下，可视化模式为“transformed”，您将看到经由数据流水线变换过后的图像和标注：

如果您只想可视化原始数据集，只需将模式设置为“original”：

```
python tools/visualizations/browse_dataset.py configs/textdet/dbnet/dbnet_resnet50-
↪dcnv2_fpnc_1200e_icdar2015.py -m original
```

或者，您也可以使用“pipeline”模式来可视化整个数据流水线的中间结果：

```
python tools/visualizations/browse_dataset.py configs/textdet/dbnet/dbnet_resnet50-
↪dcnv2_fpnc_1200e_icdar2015.py -m pipeline
```

另外，用户还可以通过指定数据集配置文件的路径来可视化数据集的原始图像及其对应的标注，例如：

```
python tools/visualizations/browse_dataset.py configs/textrecog/_base_/datasets/
↪icdar2015.py
```

部分数据集可能有多个变体。例如，`icdar2015` 文本识别数据集的配置文件中包含两个测试集变体，分别为 `icdar2015_textrecog_test` 和 `icdar2015_1811_textrecog_test`，如下所示：

```
icdar2015_textrecog_test = dict(
    ann_file='textrecog_test.json',
    # ...
    )

icdar2015_1811_textrecog_test = dict(
    ann_file='textrecog_test_1811.json',
    # ...
)
```

在这种情况下，用户可以通过指定 `-p` 参数来可视化不同的变体，例如，使用以下命令可视化 `icdar2015_1811_textrecog_test` 变体：

```
python tools/visualizations/browse_dataset.py configs/textrecog/_base_/datasets/
↪icdar2015.py -p icdar2015_1811_textrecog_test
```

基于该工具，用户可以轻松地查看数据集的原始图像及其对应的标注，以便于检查数据集的标注是否正确。

10.1.2 优化器参数策略可视化工具

MMOCR 提供了优化器参数可视化工具 `tools/visualizations/vis_scheduler.py` 以辅助用户排查优化器的超参数调度器（无需训练），支持学习率（learning rate）和动量（momentum）。

工具简介

```
python tools/visualizations/vis_scheduler.py \
    ${CONFIG_FILE} \
    [-p, --parameter ${PARAMETER_NAME}] \
    [-d, --dataset-size ${DATASET_SIZE}] \
    [-n, --ngpus ${NUM_GPUs}] \
    [-s, --save-path ${SAVE_PATH}] \
    [--title ${TITLE}] \
    [--style ${STYLE}] \
    [--window-size ${WINDOW_SIZE}] \
    [--cfg-options]
```

所有参数的说明:

- `config` : 模型配置文件的路径。
- **`-p, parameter`**: 可视化参数名，只能为 `["lr", "momentum"]` 之一，默认为 `"lr"`.
- **`-d, --dataset-size`**: 数据集的大小。如果指定，`build_dataset` 将被跳过并使用这个大小作为数据集大小，默认使用 `build_dataset` 所得数据集的大小。
- **`-n, --ngpus`**: 使用 GPU 的数量, 默认为 1。
- **`-s, --save-path`**: 保存的可视化图片的路径，默认不保存。
- `--title`: 可视化图片的标题，默认为配置文件名。
- `--style`: 可视化图片的风格，默认为 `whitegrid`。
- `--window-size`: 可视化窗口大小，如果没有指定，默认为 `12*7`。如果需要指定，按照格式 'W*H'。
- `--cfg-options`: 对配置文件的修改，参考学习配置文件。

注解： 部分数据集在解析标注阶段比较耗时，可直接将 `-d, dataset-size` 指定数据集的大小，以节约时间。

如何在开始训练前可视化学习率曲线

你可以使用如下命令来绘制配置文件 configs/textdet/dbnet/dbnet_resnet50-dcnv2_fpnc_1200e_icdar2015.py 将会使用的变化率曲线：

```
python tools/visualizations/vis_scheduler.py configs/textdet/dbnet/dbnet_resnet50-
↪dcnv2_fpnc_1200e_icdar2015.py -d 100
```

10.2 分析工具

10.2.1 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。

10.2.2 计算 FLOPs 和参数量

我们提供一个计算 FLOPs 和参数量的方法，首先我们使用以下命令安装依赖。

```
pip install fvcore
```

计算 FLOPs 和参数量的脚本使用方法如下：

```
python tools/analysis_tools/get_flops.py ${config} --shape ${IMAGE_SHAPE}
```

获取 dbnet_resnet18_fpnc_100k_synthtext.py FLOPs 和参数量的示例命令如下。

```
python tools/analysis_tools/get_flops.py configs/textdet/dbnet/dbnet_resnet18_fpnc_
↪100k_synthtext.py --shape 1024 1024
```

输出如下:

```
input shape is  (1, 3, 1024, 1024)
| module                     | #parameters or shape | #flops  |
| :------------------------- | :------------------- | :------ |
| model                      | 12.341M              | 63.955G |
| backbone                   | 11.177M              | 38.159G |
| backbone.conv1             | 9.408K               | 2.466G  |
| backbone.conv1.weight      | (64, 3, 7, 7)        |         |
| backbone.bn1               | 0.128K               | 83.886M |
| backbone.bn1.weight        | (64,)                |         |
| backbone.bn1.bias          | (64,)                |         |
| backbone.layer1            | 0.148M               | 9.748G  |
| backbone.layer1.0          | 73.984K              | 4.874G  |
| backbone.layer1.1          | 73.984K              | 4.874G  |
| backbone.layer2            | 0.526M               | 8.642G  |
| backbone.layer2.0          | 0.23M                | 3.79G   |
| backbone.layer2.1          | 0.295M               | 4.853G  |
| backbone.layer3            | 2.1M                 | 8.616G  |
| backbone.layer3.0          | 0.919M               | 3.774G  |
| backbone.layer3.1          | 1.181M               | 4.842G  |
| backbone.layer4            | 8.394M               | 8.603G  |
| backbone.layer4.0          | 3.673M               | 3.766G  |
| backbone.layer4.1          | 4.721M               | 4.837G  |
| neck                       | 0.836M               | 14.887G |
| neck.lateral_convs         | 0.246M               | 2.013G  |
| neck.lateral_convs.0.conv  | 16.384K              | 1.074G  |
| neck.lateral_convs.1.conv  | 32.768K              | 0.537G  |
| neck.lateral_convs.2.conv  | 65.536K              | 0.268G  |
| neck.lateral_convs.3.conv  | 0.131M               | 0.134G  |
| neck.smooth_convs          | 0.59M                | 12.835G |
| neck.smooth_convs.0.conv   | 0.147M               | 9.664G  |
| neck.smooth_convs.1.conv   | 0.147M               | 2.416G  |
| neck.smooth_convs.2.conv   | 0.147M               | 0.604G  |
| neck.smooth_convs.3.conv   | 0.147M               | 0.151G  |
| det_head                   | 0.329M               | 10.909G |
| det_head.binarize          | 0.164M               | 10.909G |
| det_head.binarize.0        | 0.147M               | 9.664G  |
| det_head.binarize.1        | 0.128K               | 20.972M |
| det_head.binarize.3        | 16.448K              | 1.074G  |
```

(下页继续)

(续上页)

```
| det_head.binarize.4       | 0.128K              | 83.886M |
| det_head.binarize.6       | 0.257K              | 67.109M |
| det_head.threshold        | 0.164M              |         |
| det_head.threshold.0      | 0.147M              |         |
| det_head.threshold.1      | 0.128K              |         |
| det_head.threshold.3      | 16.448K             |         |
| det_head.threshold.4      | 0.128K              |         |
| det_head.threshold.6      | 0.257K              |         |
!!!Please be cautious if you use the results in papers. You may need to check if all␣
↪ops are supported and verify that the flops computation is correct.
```

CHAPTER 11

数据元素与数据结构

MMOCR 基于 MMEngine: 抽象数据接口 将各任务所需的数据统一封装入 `data_sample` 中。MMEngine 的抽象数据接口实现了基础的增/删/改/查功能，且支持不同设备间的数据迁移，也支持了类字典和张量的操作，充分满足了数据的日常使用需求，这也使得不同算法的数据接口可以得到统一。

得益于统一的数据封装，算法库内的*`visualizer`*，*`evaluator`*，*`dataset`* 等各个模块间的数据流通都得到了极大的简化。在 MMOCR 中，我们对数据接口类型作出以下约定：

- **xxxData**: 单一粒度的数据标注或模型输出。目前 MMEngine 内置了三种粒度的数据元素，包括实例级数据（`InstanceData`），像素级数据（`PixelData`）以及图像级的标签数据（`LabelData`）。在 MMOCR 目前支持的任务中，文本检测以及关键信息抽取任务使用 `InstanceData` 来封装文本实例的检测框及对应标签，而文本识别任务则使用了 `LabelData` 来封装文本内容。
- **xxxDataSample**: 继承自 MMEngine: 数据基类 `BaseDataElement`，用于保存单个任务的训练或测试样本的**所有**标注及预测信息。如文本检测任务的数据样本类*`TextDetDataSample`*，文本识别任务的数据样本类*`TextRecogDataSample`*，以及关键信息抽任务的数据样本类*`KIEDataSample`*。

下面，我们将分别介绍数据元素 **xxxData** 与数据样本 **xxxDataSample** 在 MMOCR 中的实际应用。

11.1 数据元素 xxxData

InstanceData 和 LabelData 是 MMEngine 中定义的基础数据元素，用于封装不同粒度的标注数据或模型输出。在 MMOCR 中，我们针对不同任务中实际使用的数据类型，分别采用了 InstanceData 与 LabelData 进行了封装。

11.1.1 InstanceData

在**文本检测**任务中，检测器关注的是实例级别的文字样本，因此我们使用 InstanceData 来封装该任务所需的数据。其所需的训练标注和预测输出通常包含了矩形或多边形边界盒，以及边界盒标签。由于文本检测任务只有一种正样本类，即 “text”，在 MMOCR 中我们默认使用 0 来编号该类别。以下代码示例展示了如何使用 InstanceData 数据抽象接口来封装文本检测任务中使用的数据类型。

```
import torch
from mmengine.structures import InstanceData

# 定义 gt_instance 用于封装边界盒的标注信息
gt_instance = InstanceData()
gt_instance.bbox = torch.Tensor([[0, 0, 10, 10], [10, 10, 20, 20]])
gt_instance.polygons = torch.Tensor([[[0, 0], [10, 0], [10, 10], [0, 10]],
                                     [[10, 10], [20, 10], [20, 20], [10, 20]]])
gt_instance.label = torch.Tensor([0, 0])

# 定义 pred_instance 用于封装模型的输出信息
pred_instances = InstanceData()
pred_polygons, scores = model(input)
pred_instances.polygons = pred_polygons
pred_instances.scores = scores
```

MMOCR 中对 InstanceData 字段的约定如下表所示。值得注意的是，InstanceData 中的各字段的长度必须为与样本中的实例个数 N 相等。

11.1.2 LabelData

对于**文字识别**任务，标注内容和预测内容都会使用 LabelData 进行封装。

```
import torch
from mmengine.data import LabelData

# 定义一个 gt_text 用于封装标签文本内容
gt_text = LabelData()
gt_text.item = 'MMOCR'
```

(下页继续)

(续上页)

```
# 定义一个 pred_text 对象用于封装预测文本以及置信度
pred_text = LabelData()
index, score = model(input)
text = dictionary.idx2str(index)
pred_text.score = score
pred_text.item = text
```

MMOCR 中对 LabelData 字段的约定如下表所示：

11.2 数据样本 xxxDataSample

通过定义统一的数据结构，我们可以方便地将标注数据和预测结果进行统一封装，使代码库不同模块间的数据传递更加便捷。在 MMOCR 中，我们基于现在支持的三个任务及其所需要的数据分别封装了三种数据抽象，包括文本检测任务数据抽象*TextDetDataSample*，文本识别任务数据抽象*TextRecogDataSample*，以及关键信息抽取任务数据抽象*KIEDataSample*。这些数据抽象均继承自 MMEngine: 数据基类 BaseDataElement，用于保存单个任务的训练或测试样本的所有标注及预测信息。

11.2.1 文本检测任务数据抽象 TextDetDataSample

TextDetDataSample 用于封装文字检测任务所需的数据，其主要包含了两个字段 gt_instances 与 pred_instances，分别用于存放标注信息与预测结果。

其中会用到的*InstanceData* 约定字段有：

由于文本检测模型通常只会输出 bboxes/polygons 中的一项，因此我们只需确保这两项中的一个被赋值即可。

以下示例代码展示了 TextDetDataSample 的使用方法：

```
import torch
from mmengine.data import TextDetDataSample

data_sample = TextDetDataSample()
# 指定当前图片的标注信息
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))
gt_instances.labels = torch.zeros((5,), dtype=torch.long)
data_sample.gt_instances = gt_instances

# 指定当前图片的预测信息
pred_instances = InstanceData()
pred_instances.bboxes = torch.rand((5, 4))
```

(下页继续)

(续上页)

```
pred_instances.labels = torch.zeros((5,), dtype=torch.long)
data_sample.pred_instances = pred_instances
```

11.2.2 文本识别任务数据抽象 TextRecogDataSample

TextRecogDataSample 用于封装文字识别任务的数据。它有两个属性，gt_text 和 pred_text，分别用于存放标注信息和预测结果。

以下示例代码展示了*TextRecogDataSample*的使用方法：

```
import torch
from mmengine.data import TextRecogDataSample

data_sample = TextRecogDataSample()
# 指定当前图片的标注信息
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

# 指定当前图片的预测结果
pred_text = LabelData(metainfo=img_meta)
pred_text.item = 'mmocr'
data_sample.pred_text = pred_text
```

其中会用到的 LabelData 字段有：

11.2.3 关键信息抽取任务数据抽象 KIEDataSample

KIEDataSample 用于封装 KIE 任务所需的数据，其同样约定了两个属性，即 gt_instances 与 pred_instances，分别用于存放标注信息与预测结果。

该任务会用到的*InstanceData*字段如下表所示：

警告： 由于 KIE 任务的模型实现尚未有统一标准，该设计目前仅考虑了 SDMGR 模型的使用场景。因此，该设计有可能在我们支持更多 KIE 模型后产生变动。

以下示例代码展示了*KIEDataSample*的使用方法。

```
import torch
from mmengine.data import KIEDataSample
```

(下页继续)

(续上页)

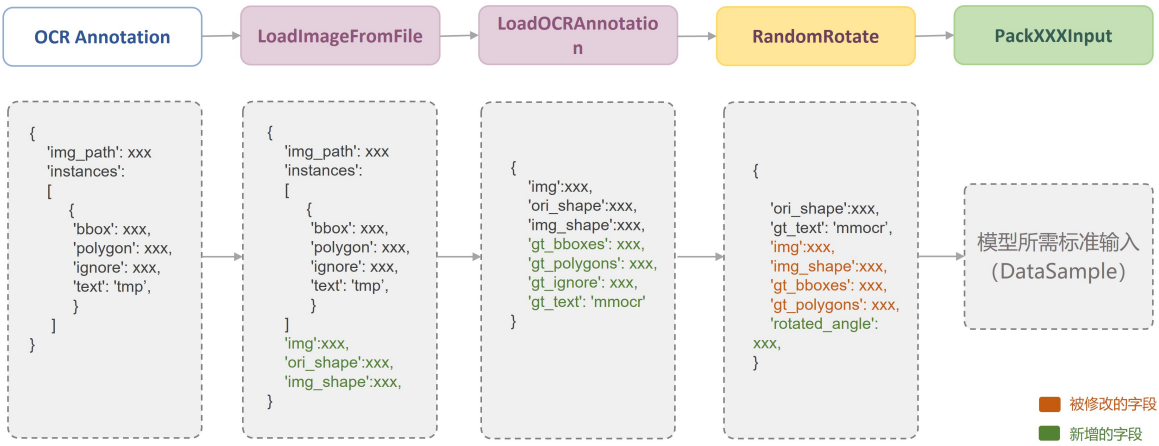
```
data_sample = KIEDataSample()
# 指定当前图片的标注信息
img_meta = dict(img_shape=(800, 1196, 3),pad_shape=(800, 1216, 3))
gt_instances = InstanceData(metainfo=img_meta)
gt_instances.bboxes = torch.rand((5, 4))
gt_instances.labels = torch.zeros((5,), dtype=torch.long)
gt_instances.texts = ['text1', 'text2', 'text3', 'text4', 'text5']
gt_instances.edge_lebels = torch.randint(-1, 2, (5, 5))
data_sample.gt_instances = gt_instances

# 指定当前图片的预测信息
pred_instances = InstanceData()
pred_instances.bboxes = torch.rand((5, 4))
pred_instances.labels = torch.rand((5,))
pred_instances.edge_labels = torch.randint(-1, 2, (10, 10))
pred_instances.edge_scores = torch.rand((10, 10))
data_sample.pred_instances = pred_instances
```

CHAPTER 12

数据变换与流水线

在 MMOCR 的设计中，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预处理、数据增强、数据格式化等相关功能。目前，如下表所示，MMOCR 中共实现了 5 类数据变换方法：

由于每一个数据变换类之间都是相互独立的，因此，在约定好固定的数据存储字段后，我们可以便捷地采用任意的数据变换组合来构建数据流水线（Pipeline）。如下图所示，在 MMOCR 中，一个典型的训练数据流水线主要由**数据读取**、**图像增强**以及**数据格式化**三部分构成，用户只需要在配置文件中定义相关的数据流水线列表，并指定具体所需的数据变换类及其参数即可：




```
train_pipeline_r18 = [
    # 数据读取（图像）
```

(下页继续)

(续上页)

```
    dict(
        type='LoadImageFromFile',
        color_type='color_ignore_orientation'),
    # 数据读取（标注）
    dict(
        type='LoadOCRAnnotations',
        with_polygon=True,
        with_bbox=True,
        with_label=True,
    ),
    # 使用 ImgAug 作数据增强
    dict(
        type='ImgAugWrapper',
        args=[['Fliplr', 0.5],
              dict(cls='Affine', rotate=[-10, 10]), ['Resize', [0.5, 3.0]]]),
    # 使用 MMOCR 内置的图像增强
    dict(type='RandomCrop', min_side_ratio=0.1),
    dict(type='Resize', scale=(640, 640), keep_ratio=True),
    dict(type='Pad', size=(640, 640)),
    # 数据格式化
    dict(
        type='PackTextDetInputs',
        meta_keys=('img_path', 'ori_shape', 'img_shape'))
]
```

小技巧： 更多有关数据流水线配置的教程可见配置文档。下面，我们将简单介绍 MMOCR 中已支持的数据变换类型。

对于每一个数据变换方法，MMOCR 都严格按照文档字符串（docstring）规范在源码中提供了详细的代码注释。例如，每一个数据转换类的头部我们都注释了“需求字段”（Required keys），“修改字段”（Modified Keys）与“添加字段”（Added Keys）。其中，“需求字段”代表该数据转换方法对于输入数据所需包含字段的强制需求，而“修改字段”与“添加字段”则表明该方法可能会在原有数据基础之上修改或添加的字段。例如，LoadImageFromFile 实现了图片的读取功能，其需求字段为图像的存储路径 img_path，而修改字段则包括了读入的图像信息 img，以及图片当前尺寸 img_shape，图片原始尺寸 ori_shape 等图片属性。

```
@TRANSFORMS.register_module()
class LoadImageFromFile(MMCV_LoadImageFromFile):
    # 在每一个数据变换方法的头部，我们都提供了详细的代码注释。
    """Load an image from file.

    Required Keys:
```

(下页继续)

(续上页)

```
    - img_path

    Modified Keys:

    - img
    - img_shape
    - ori_shape
    """
```

注解： 在 MMOCR 的数据流水线中，图像及标签等信息被统一保存在字典中。通过统一的字段名，我们可以在不同的数据变换方法间灵活地传递数据。因此，了解 MMOCR 中常用的约定字段名是非常重要的。

为方便用户查询，下表列出了 MMOCR 中各数据转换（Data Transform）类常用的字段约定和说明。

12.1 数据读取 - loading.py

数据读取类主要实现了不同文件格式、后端读取图片及加载标注信息的功能。目前，MMOCR 内部共实现了以下数据读取类的 Data Transforms：

12.2 数据增强 - xxx_transforms.py

数据增强是文本检测、识别等任务中必不可少的流程之一。目前，MMOCR 中共实现了数十种文本领域内常用的数据增强模块，依据其任务类型，分别为通用 OCR 数据增强模块 ocr_transforms.py，文本检测数据增强模块 textdet_transforms.py，以及文本识别数据增强模块 textrecog_transforms.py。

具体而言，`ocr_transforms.py` 中实现了随机剪裁、随机旋转等各任务通用的数据增强模块：

`textdet_transforms.py` 则实现了文本检测任务中常用的数据增强模块：

`textrecog_transforms.py` 中实现了文本识别任务中常用的数据增强模块：

警告： 以上表格仅选择性地对部分数据增强方法作简要介绍，更多数据增强方法介绍请参考 API 文档或阅读代码内的文档注释。

12.3 数据格式化 - formatting.py

数据格式化负责将图像、真实标签以及其它常用信息等打包成一个字典。不同的任务通常依赖于不同的数据格式化数据变换类。例如:

12.4 跨库数据适配器 - adapters.py

跨库数据适配器打通了 MMOCR 与其他 OpenMMLab 系列算法库如 MMDetection 之间的数据格式，使得跨项目调用其它开源算法库的配置文件及算法成为了可能。目前，MMOCR 实现了 `MMDet2MMOCR` 以及 `MMOCR2MMDet`，使得数据可以在 MMDetection 与 MMOCR 的格式之间自由转换；借助这些适配转换器，用户可以在 MMOCR 算法库内部轻松调用任何 MMDetection 已支持的检测算法，并在 OCR 相关数据集上进行训练。例如，我们以 Mask R-CNN 为例提供了教程，展示了如何在 MMOCR 中使用 MMDetection 的检测算法训练文本检测器。

12.5 包装类 - wrappers.py

为了方便用户在 MMOCR 内部无缝调用常用的 CV 算法库，我们在 wrappers.py 中提供了相应的包装类。其主要打通了 MMOCR 与其它第三方算法库之间的数据格式和转换标准，使得用户可以在 MMOCR 的配置文件内直接配置使用这些第三方库提供的数据变换方法。目前支持的包装类有:

12.5.1 `ImgAugWrapper` 示例

例如，在原生的 ImgAug 中，我们可以按照如下代码定义一个 `Sequential` 类型的数据增强流程，对图像分别进行随机翻转、随机旋转和随机缩放:

```
import imgaug.augmenters as iaa

aug = iaa.Sequential(
  iaa.Fliplr(0.5),                # 以概率 0.5 进行水平翻转
  iaa.Affine(rotate=(-10, 10)),   # 随机旋转 -10 到 10 度
  iaa.Resize((0.5, 3.0))          # 随机缩放到 50% 到 300% 的尺寸
)
```

而在 MMOCR 中，我们可以通过 `ImgAugWrapper` 包装类，将上述数据增强流程直接配置到 `train_pipeline` 中:

```
dict(
  type='ImgAugWrapper',
  args=[
    ['Fliplr', 0.5],
```

(下页继续)

(续上页)

```
    dict(cls='Affine', rotate=[-10, 10]),
    ['Resize', [0.5, 3.0]],
  ]
)
```

其中，args 参数接收一个列表，列表中的每个元素可以是一个列表，也可以是一个字典。如果是列表，则列表的第一个元素为 imgaug.augmenters 中的类名，后面的元素为该类的初始化参数；如果是字典，则字典的 cls 键对应 imgaug.augmenters 中的类名，其他键值对则对应该类的初始化参数。

12.5.2 TorchVisionWrapper 示例

例如，在原生的 TorchVision 中，我们可以按照如下代码定义一个 Compose 类型的数据变换流程，对图像进行色彩抖动：

```
import torchvision.transforms as transforms

aug = transforms.Compose([
  transforms.ColorJitter(
    brightness=32.0 / 255,  # 亮度抖动范围
    saturation=0.5)         # 饱和度抖动范围
])
```

而在 MMOCR 中，我们可以通过 TorchVisionWrapper 包装类，将上述数据变换流程直接配置到 train_pipeline 中：

```
dict(
  type='TorchVisionWrapper',
  op='ColorJitter',
  brightness=32.0 / 255,
  saturation=0.5
)
```

其中，op 参数为 torchvision.transforms 中的类名，后面的参数则对应该类的初始化参数。

CHAPTER 13

模型评测

注解： 阅读此文档前，建议您先了解 MMEngine: 模型精度评测基本概念。

13.1 评测指标

MMOCR 基于 MMEngine: BaseMetric 基类实现了常用的文本检测、文本识别以及关键信息抽取任务的评测指标，用户可以通过修改配置文件中的 `val_evaluator` 与 `test_evaluator` 字段来便捷地指定验证与测试阶段采用的评测方法。例如，以下配置展示了如何在文本检测算法中使用 `HmeanIOUMetric` 来评测模型性能。

```
# 文本检测任务中通常使用 HmeanIOUMetric 来评测模型性能
val_evaluator = [dict(type='HmeanIOUMetric')]

# 此外，MMOCR 也支持相同任务下的多种指标组合评测，如同时使用 WordMetric 及 CharMetric
val_evaluator = [
    dict(type='WordMetric', mode=['exact', 'ignore_case', 'ignore_case_symbol']),
    dict(type='CharMetric')
]
```

小技巧： 更多评测相关配置请参考评测配置教程。

如下表所示，MMOCR 目前针对文本检测、识别、及关键信息抽取等任务共内置了 5 种评测指标，分别为 `HmeanIOUMetric`，`WordMetric`，`CharMetric`，`OneMinusNEDMetric`，和 `F1Metric`。

通常来说，每一类任务所采用的评测标准是约定俗成的，用户一般无须深入了解或手动修改评测方法的内部实现。然而，为了方便用户实现更加定制化的需求，本文档将进一步介绍了 MMOCR 内置评测算法的具体实现策略，以及可配置参数。

13.1.1 HmeanIOUMetric

HmeanIOUMetric 是文本检测任务中应用最广泛的评测指标之一，因其计算了检测精度（Precision）与召回率（Recall）之间的调和平均数（Harmonic mean, H-mean），故得名 `HmeanIOUMetric`。记精度为 P，召回率为 R，则 `HmeanIOUMetric` 可由下式计算得到：

$$H = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P + R}$$

另外，由于其等价于 $\beta = 1$ 时的 F-score (又称 F-measure 或 F-metric)，`HmeanIOUMetric` 有时也被写作 `F1Metric` 或 `f1-score` 等：

$$F_1 = (1 + \beta^2) \cdot \frac{PR}{\beta^2 \cdot P + R} = \frac{2PR}{P + R}$$

在 MMOCR 的设计中，`HmeanIOUMetric` 的计算可以概括为以下几个步骤：

1. 过滤无效的预测边界盒

   - 依据置信度阈值 `pred_score_thrs` 过滤掉得分较低的预测边界盒
   - 依据 `ignore_precision_thr` 阈值过滤掉与 `ignored` 样本重合度过高的预测边界盒

   值得注意的是，`pred_score_thrs` 默认将**自动搜索**一定范围内的**最佳阈值**，用户也可以通过手动修改配置文件来自定义搜索范围：

   ```
   # HmeanIOUMetric 默认以 0.1 为步长搜索 [0.3, 0.9] 范围内的最佳得分阈值
   val_evaluator = dict(type='HmeanIOUMetric', pred_score_thrs=dict(start=0.3,␣
   ↪stop=0.9, step=0.1))
   ```

2. 计算 IoU 矩阵

   - 在数据处理阶段，`HmeanIOUMetric` 会计算并维护一个 $M \times N$ 的 IoU 矩阵 `iou_metric`，以方便后续的边界盒配对步骤。其中，M 和 N 分别为标签边界盒与过滤后预测边界盒的数量。由此，该矩阵的每个元素都存放了第 m 个标签边界盒与第 n 个预测边界盒之间的交并比（IoU）。

3. 基于相应的配对策略统计能被准确匹配的 GT 样本数

   尽管 `HmeanIOUMetric` 可以由固定的公式计算取得，不同的任务或算法库内部的具体实现仍可能存在一些细微差别。这些差异主要体现在采用不同的策略来匹配真实与预测边界盒，从而导致最终得分的差距。目前，MMOCR 内部的 `HmeanIOUMetric` 共支持两种不同的匹配策略，即 `vanilla` 与 `max_matching`。如下所示，用户可以通过修改配置文件来指定不同的匹配策略。

- vanilla 匹配策略

  HmeanIOUMetric 默认采用 vanilla 匹配策略，该实现与 MMOCR 0.x 版本中的 hmean-iou 及 ICDAR 系列**官方文本检测竞赛的评测标准保持一致**，采用先到先得的匹配方式对标签边界盒（Ground-truth bbox）与预测边界盒（Predicted bbox）进行配对。

```
# 不指定 strategy 时，HmeanIOUMetric 默认采用 'vanilla' 匹配策略
val_evaluator = dict(type='HmeanIOUMetric')
```

- max_matching 匹配策略

  针对现有匹配机制中的不完善之处，MMOCR 算法库实现了一套更高效的匹配策略，用以最大化匹配数目。

```
# 指定采用 'max_matching' 匹配策略
val_evaluator = dict(type='HmeanIOUMetric', strategy='max_matching')
```

注解： 我们建议面向学术研究的开发用户采用默认的 vanilla 匹配策略，以保证与其他论文的对比结果保持一致。而面向工业应用的开发用户则可以采用 max_matching 匹配策略，以获得精准的结果。

4. 根据上文介绍的 HmeanIOUMetric 公式计算最终的评测得分

13.1.2 WordMetric

WordMetric 实现了**单词级别**的文本识别评测指标，并内置了 exact，ignore_case，及 ignore_case_symbol 三种文本匹配模式，用户可以在配置文件中修改 mode 字段来自由组合输出一种或多种文本匹配模式下的 WordMetric 得分。

```
# 在文本识别任务中使用 WordMetric 评测
val_evaluator = [
    dict(type='WordMetric', mode=['exact', 'ignore_case', 'ignore_case_symbol'])
]
```

- exact：全匹配模式，即，预测与标签完全一致才能被记录为正确样本。
- ignore_case：忽略大小写的匹配模式。
- ignore_case_symbol：忽略大小写及符号的匹配模式，这也是大部分学术论文中报告的文本识别准确率；MMOCR 报告的识别模型性能默认采用该匹配模式。

假设真实标签为 MMOCR!，模型的输出结果为 mmocr，则三种匹配模式下的 WordMetric 得分分别为：{'exact': 0, 'ignore_case': 0, 'ignore_case_symbol': 1}。

13.1.3 CharMetric

CharMetric 实现了**不区分大小写**的**字符级别**的文本识别评测指标。

```
# 在文本识别任务中使用 CharMetric 评测
val_evaluator = [dict(type='CharMetric')]
```

具体而言，CharMetric 会输出两个评测评测指标，即字符精度 char_precision 和字符召回率 char_recall。设正确预测的字符（True Positive）数量为 σ_{tp}，则精度 *P* 和召回率 *R* 可由下式计算取得：

$$P = \frac{\sigma_{tp}}{\sigma_{pred}}, R = \frac{\sigma_{tp}}{\sigma_{gt}}$$

其中，σ_{gt} 与 σ_{pred} 分别为标签文本与预测文本所包含的字符总数。

例如，假设标签文本为“MM**O**CR”，预测文本为“mm**0**cR**1**”，则使用 CharMetric 评测指标的得分为：

$$P = \frac{4}{6}, R = \frac{4}{5}$$

13.1.4 OneMinusNEDMetric

OneMinusNEDMetric(1-N.E.D) 常用于中文或英文**文本行级别**标注的文本识别评测，不同于全匹配的评测标准要求预测与真实样本完全一致，该评测指标使用归一化的编辑距离（Edit Distance，又名莱温斯坦距离 Levenshtein Distance）来测量预测文本与真实文本之间的差异性，从而在评测长文本样本时能够更好地区分出模型的性能差异。假设真实和预测文本分别为 s_i 和 $\hat{s_i}$，其长度分别为 l_i 和 $\hat{l_i}$，则 OneMinusNEDMetric 得分可由下式计算得到：

$$score = 1 - \frac{1}{N}\sum_{i=1}^{N}\frac{D(s_i, \hat{s_i})}{max(l_i, \hat{l_i})}$$

其中，*N* 是样本总数，$D(s_1, s_2)$ 为两个字符串之间的编辑距离。

例如，假设真实标签为“OpenMMLabMMOCR”，模型 A 的预测结果为“0penMMLabMMOCR”，模型 B 的预测结果为“uvwxyz”，则采用全匹配和 OneMinusNEDMetric 评测指标的结果分别为:

由上表可以发现，尽管模型 A 仅预测错了一个字母，而模型 B 全部预测错误，在使用全匹配的评测指标时，这两个模型的得分都为 0；而使用 OneMinuesNEDMetric 的评测指标则能够更好地区分模型在**长文本**上的性能差异。

13.1.5 F1Metric

F1Metric 实现了针对 KIE 任务的 F1-Metric 评测指标，并提供了 micro 和 macro 两种评测模式。

```
val_evaluator = [
    dict(type='F1Metric', mode=['micro', 'macro'],
]
```

- micro 模式：依据 True Positive，False Negative，及 False Positive 总数来计算全局 F1-Metric 得分。
- macro 模式：依据类别标签计算每一类的 F1-Metric，并求平均值。

13.1.6 自定义评测指标

对于追求更高定制化功能的用户，MMOCR 也支持自定义实现不同类型的评测指标。一般来说，用户只需要新建自定义评测指标类 CustomizedMetric 并继承 MMEngine: BaseMetric，然后分别重写数据格式处理方法 process 以及指标计算方法 compute_metrics。最后，将其加入 METRICS 注册器即可实现任意定制化的评测指标。

```
from mmengine.evaluator import BaseMetric
from mmocr.registry import METRICS

@METRICS.register_module()
class CustomizedMetric(BaseMetric):

    def process(self, data_batch: Sequence[Dict], predictions: Sequence[Dict]):
    """ process 接收两个参数，分别为 data_batch 存放真实标签信息，以及 predictions
        存放预测结果。process 方法负责将标签信息转换并存放至 self.results 变量中
    """
        pass

    def compute_metrics(self, results: List):
    """ compute_metric 使用经过 process 方法处理过的标签数据计算最终评测得分
    """
        pass
```

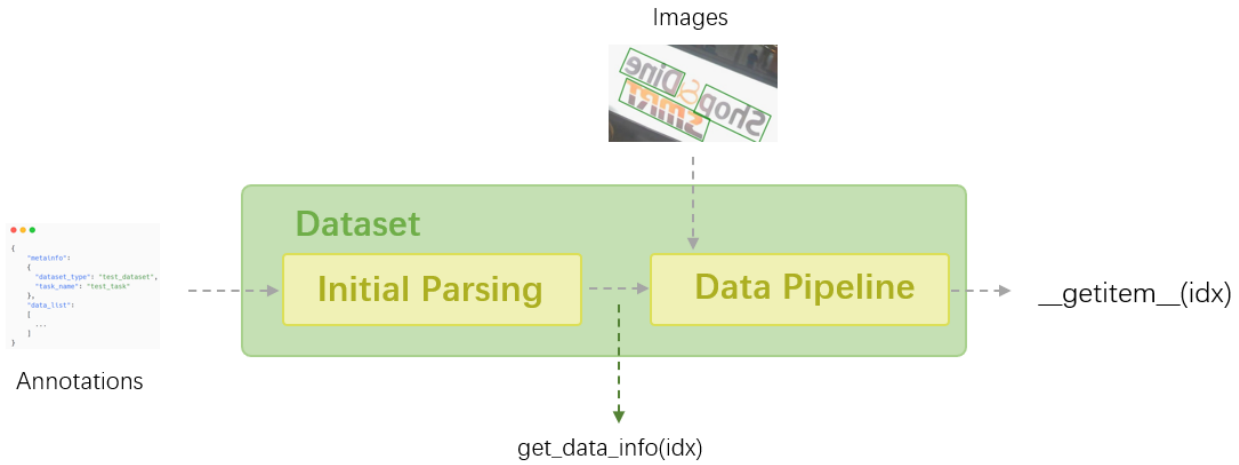
注解： 更多内容可参见 MMEngine 文档: BaseMetric。

CHAPTER 14

数据集类

14.1 概览

在 MMOCR 中，所有的数据集都通过不同的基于 mmengine.BaseDataset 的 Dataset 类进行处理。Dataset 类负责加载数据并进行初始解析，然后将其馈送到数据流水线 进行数据预处理、增强、格式化等操作。



在本教程中，我们将介绍 Dataset 类的一些常见接口，以及 MMOCR 中 Dataset 实现的使用以及它们支持的注释类型。

小技巧： Dataset 类支持一些高级功能，例如懒加载、数据序列化、利用各种数据集包装器执行数据连接、重复和类别平衡。这些内容将不在本教程中介绍，但您可以阅读 MMEngine: BaseDataset 了解更多详细信息。

14.2 常见接口

现在，让我们看一个具体的示例并学习 Dataset 类的一些典型接口。`OCRDataset` 是 MMOCR 中默认使用的 Dataset 实现，因为它的标注格式足够灵活，支持 所有 OCR 任务（详见*OCRDataset*）。现在我们将实例化一个 `OCRDataset` 对象，其中将加载 `tests/data/det_toy_dataset` 中的玩具数据集。

```
from mmocr.datasets import OCRDataset
from mmengine.registry import init_default_scope
init_default_scope('mmocr')

train_pipeline = [
    dict(
        type='LoadImageFromFile'),
    dict(
        type='LoadOCRAnnotations',
        with_polygon=True,
        with_bbox=True,
        with_label=True,
    ),
    dict(type='RandomCrop', min_side_ratio=0.1),
    dict(type='Resize', scale=(640, 640), keep_ratio=True),
    dict(type='Pad', size=(640, 640)),
    dict(
        type='PackTextDetInputs',
        meta_keys=('img_path', 'ori_shape', 'img_shape'))
]
dataset = OCRDataset(
    data_root='tests/data/det_toy_dataset',
    ann_file='textdet_test.json',
    test_mode=False,
    pipeline=train_pipeline)
```

让我们查看一下这个数据集的大小：

```
>>> print(len(dataset))

10
```

通常，Dataset 类加载并存储两种类型的信息：（1）**元信息**：储存数据集的属性，例如此数据集中可用的对象类别。（2）**标注**：图像的路径及其标签。我们可以通过 `dataset.metainfo` 访问元信息：

```
>>> from pprint import pprint
>>> pprint(dataset.metainfo)
```

(下页继续)

(续上页)

```
{'category': [{'id': 0, 'name': 'text'}],
 'dataset_type': 'TextDetDataset',
 'task_name': 'textdet'}
```

对于标注，我们可以通过 `dataset.get_data_info(idx)` 访问它。该方法返回一个字典，其中包含数据集中第 `idx` 个样本的信息。该样本已经经过初步解析，但尚未由*数据流水线* 处理。

```
>>> from pprint import pprint
>>> pprint(dataset.get_data_info(0))

{'height': 720,
 'img_path': 'tests/data/det_toy_dataset/test/img_10.jpg',
 'instances': [{'bbox': [260.0, 138.0, 284.0, 158.0],
                'bbox_label': 0,
                'ignore': True,
                'polygon': [261, 138, 284, 140, 279, 158, 260, 158]},
                ...,
               {'bbox': [1011.0, 157.0, 1079.0, 173.0],
                'bbox_label': 0,
                'ignore': True,
                'polygon': [1011, 157, 1079, 160, 1076, 173, 1011, 170]}],
 'sample_idx': 0,
 'seg_map': 'test/gt_img_10.txt',
 'width': 1280}
```

另一方面，我们可以通过 `dataset[idx]` 或 `dataset.__getitem__(idx)` 获取由数据流水线完整处理过后的样本，该样本可以直接馈入模型并执行完整的训练/测试循环。它有两个字段：

- `inputs`：经过数据增强后的图像；
- `data_samples`：包含经过数据增强后的标注和元信息的*DataSample*，这些元信息可能由一些数据变换产生，并用以记录该样本的某些关键属性。

```
>>> pprint(dataset[0])

{'data_samples': <TextDetDataSample(

    META INFORMATION
    ori_shape: (720, 1280)
    img_path: 'tests/data/det_toy_dataset/imgs/test/img_10.jpg'
    img_shape: (640, 640)

    DATA FIELDS
```

(下页继续)

(续上页)

```
    gt_instances: <InstanceData(

            META INFORMATION

            DATA FIELDS
            labels: tensor([0, 0, 0])
            polygons: [array([207.33984 , 104.65409 , 208.34634 ,  84.528305, 231.
↪49594 ,
                    86.54088 , 226.46341 , 104.65409 , 207.33984 , 104.65409 ],
                  dtype=float32), array([237.53496 , 103.6478  , 235.52196 ,  84.
↪528305, 365.36096 ,
                    86.54088 , 364.35446 , 107.67296 , 237.53496 , 103.6478  ],
                  dtype=float32), array([105.68293, 166.03773, 105.68293, 151.
↪94969, 177.14471, 150.94339,
                   178.15121, 165.03145, 105.68293, 166.03773], dtype=float32)]
            ignored: tensor([ True, False,  True])
            bboxes: tensor([[207.3398,  84.5283, 231.4959, 104.6541],
                    [235.5220,  84.5283, 365.3610, 107.6730],
                    [105.6829, 150.9434, 178.1512, 166.0377]])
        ) at 0x7f7359f04fa0>
) at 0x7f735a0508e0>,
 'inputs': tensor([[[129, 111, 131,  ...,   0,   0,   0], ...
                 [ 19,  18,  15,  ...,   0,   0,   0]]], dtype=torch.uint8)}
```

14.3 数据集类及标注格式

每个数据集实现只能加载特定格式的数据集。这里列出了所有支持的数据集类及其兼容的格式，以及一个示例配置，以演示如何在实践中使用它们。

注解： 如果您不熟悉配置系统，可以阅读 数据集配置文件。

14.3.1 OCRDataset

通常，OCR 数据集中有许多不同类型的标注，在不同的子任务（如文本检测和文本识别）中，格式也经常会有所不同。这些差异可能会导致在使用不同数据集时需要不同的数据加载代码，增加了用户的学习和维护成本。

在 MMOCR 中，我们提出了一种统一的数据集格式，可以适应 OCR 的所有三个子任务：文本检测、文本识别和端到端 OCR。这种设计最大程度地提高了数据集的一致性，允许在不同任务之间重复使用数据标注，也

使得数据集管理更加方便。考虑到流行的数据集格式并不一致，MMOCR 提供了*Dataset Preparer* 来帮助用户将其数据集转换为 MMOCR 格式。我们也十分鼓励研究人员基于此数据格式开发自己的数据集。

标注格式

此标注文件是一个 `.json` 文件，存储一个包含 `metainfo` 和 `data_list` 的 `dict`，前者包括有关数据集的基本信息，后者由每个图片的标注组成。这里呈现了标注文件中的所有字段的列表，但其中某些字段仅会在特定任务中被用到。

```
{
    "metainfo":
    {
      "dataset_type": "TextDetDataset",  # 可选项: TextDetDataset/TextRecogDataset/
↪TextSpotterDataset
      "task_name": "textdet",  #  可选项: textdet/textspotter/textrecog
      "category": [{"id": 0, "name": "text"}]  # 在 textdet/textspotter 里用到
    },
    "data_list":
    [
      {
        "img_path": "test_img.jpg",
        "height": 604,
        "width": 640,
        "instances":  # 一图内的多个实例
        [
          {
            "bbox": [0, 0, 10, 20],  # textdet/textspotter 内用到, [x1, y1, x2, y2]。
            "bbox_label": 0,  # 对象类别, 在 MMOCR 中恒为 0 (文本)
            "polygon": [0, 0, 0, 10, 10, 20, 20, 0], # textdet/textspotter 内用到。 [x1,
↪ y1, x2, y2, ....]
            "text": "mmocr",  # textspotter/textrecog 内用到
            "ignore": False # textspotter/textdet 内用到, 决定是否在训练时忽略该实例
          },
          #...
        ],
      }
      #... 多图片
    ]
}
```

示例配置

以下是配置的一部分，我们在 `train_dataloader` 中使用 `OCRDataset` 加载用于文本检测模型的 ICDAR2015 数据集。请注意，`OCRDataset` 可以加载由 Dataset Preparer 准备的任何 OCR 数据集。也就是说，您可以将其用于文本识别和文本检测，但您仍然需要根据不同任务的需求修改 `pipeline` 中的数据变换。

```
pipeline = [
    dict(
        type='LoadImageFromFile'),
    dict(
        type='LoadOCRAnnotations',
        with_polygon=True,
        with_bbox=True,
        with_label=True,
    ),
    dict(
        type='PackTextDetInputs',
        meta_keys=('img_path', 'ori_shape', 'img_shape'))
]

icdar2015_textdet_train = dict(
    type='OCRDataset',
    data_root='data/icdar2015',
    ann_file='textdet_train.json',
    filter_cfg=dict(filter_empty_gt=True, min_size=32),
    pipeline=pipeline)

train_dataloader = dict(
    batch_size=16,
    num_workers=8,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=True),
    dataset=icdar2015_textdet_train)
```

14.3.2 RecogLMDBDataset

当数据量非常大时，从文件中读取图像或标签可能会很慢。此外，在学术界，大多数场景文本识别数据集的图像和标签都以 lmdb 格式存储。(示例)

为了更接近主流实践并提高数据存储效率，MMOCR 支持通过 `RecogLMDBDataset` 从 lmdb 数据集加载图像和标签。

标注格式

MMOCR 会读取 lmdb 数据集中的以下键：

- `num_samples`：描述数据集的数据量的参数。
- 图像和标签的键分别以 `image-000000001` 和 `label-000000001` 的格式命名，索引从 1 开始。

MMOCR 在 `tests/data/rec_toy_dataset/imgs.lmdb` 中提供了一个 toy lmdb 数据集。您可以使用以下代码片段了解其格式。

```
>>> import lmdb
>>>
>>> env = lmdb.open('tests/data/rec_toy_dataset/imgs.lmdb')
>>> txn = env.begin()
>>> for k, v in txn.cursor():
>>>     print(k, v)

b'image-000000001' b'\xff...'
b'image-000000002' b'\xff...'
b'image-000000003' b'\xff...'
b'image-000000004' b'\xff...'
b'image-000000005' b'\xff...'
b'image-000000006' b'\xff...'
b'image-000000007' b'\xff...'
b'image-000000008' b'\xff...'
b'image-000000009' b'\xff...'
b'image-000000010' b'\xff...'
b'label-000000001' b'GRAND'
b'label-000000002' b'HOTEL'
b'label-000000003' b'HOTEL'
b'label-000000004' b'PACIFIC'
b'label-000000005' b'03/09/2009'
b'label-000000006' b'ANING'
b'label-000000007' b'Virgin'
b'label-000000008' b'america'
b'label-000000009' b'ATTACK'
b'label-000000010' b'DAVIDSON'
b'num-samples' b'10'
```

示例配置

以下是示例配置的一部分，我们在其中使用 `RecogLMDBDataset` 加载 toy 数据集。由于 `RecogLMDBDataset` 会将图像加载为 numpy 数组，因此如果要在数据管道中成功加载图像，应该记得把 `LoadImageFromFile` 替换成 `LoadImageFromNDArray` 。

```
pipeline = [
    dict(
        type='LoadImageFromNDArray'),
    dict(
        type='LoadOCRAnnotations',
        with_text=True,
    ),
    dict(
        type='PackTextRecogInputs',
        meta_keys=('img_path', 'ori_shape', 'img_shape'))
]

toy_textrecog_train = dict(
    type='RecogLMDBDataset',
    data_root='tests/data/rec_toy_dataset/',
    ann_file='imgs.lmdb',
    pipeline=pipeline)

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

14.3.3 RecogTextDataset

在 MMOCR 1.0 之前，MMOCR 0.x 的文本识别任务的输入是文本文件。这些格式已在 MMOCR 1.0 中弃用，这个类随时可能被删除。更多信息

标注格式

文本文件可以是 txt 格式或 jsonl 格式。简单的 .txt 标注通过空格将图像名称和词语标注分隔开，因此这种格式并无法处理文本实例中包含空格的情况。

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

jsonl 格式使用类似字典的结构来表示标注，其中键 filename 和 text 存储图像名称和单词标签。

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

示例配置

以下是一个示例配置，我们在训练中使用 RecogTextDataset 加载 txt 标签，而在测试中使用 jsonl 标签。

```
pipeline = [
    dict(
        type='LoadImageFromFile'),
    dict(
        type='LoadOCRAnnotations',
        with_polygon=True,
        with_bbox=True,
        with_label=True,
    ),
    dict(
        type='PackTextDetInputs',
        meta_keys=('img_path', 'ori_shape', 'img_shape'))
]

 # loading 0.x txt format annos
 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=pipeline)
```

(下页继续)

(续上页)

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

 # loading 0.x json line format annos
 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=pipeline))

test_dataloader = dict(
    batch_size=16,
    num_workers=8,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=False),
    dataset=jsonl_dataset)
```

14.3.4 IcdarDataset

在 MMOCR 1.0 之前，MMOCR 0.x 的文本检测输入采用了类似 COCO 格式的注释。这些格式已在 MMOCR 1.0 中弃用，这个类在将来的任何时候都可能被删除。更多信息

标注格式

```
{
  "images": [
    {
      "id": 1,
      "width": 800,
      "height": 600,
      "file_name": "test.jpg"
    }
  ],
```

(下页继续)

(续上页)

```
  "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
    }
  ]
}
```

配置示例

这是配置示例的一部分，其中我们令 `train_dataloader` 使用 `IcdarDataset` 来加载旧标签。

```
pipeline = [
    dict(
        type='LoadImageFromFile'),
    dict(
        type='LoadOCRAnnotations',
        with_polygon=True,
        with_bbox=True,
        with_label=True,
    ),
    dict(
        type='PackTextDetInputs',
        meta_keys=('img_path', 'ori_shape', 'img_shape'))
]

icdar2015_textdet_train = dict(
    type='IcdarDatasetDataset',
    data_root='data/det/icdar2015',
    ann_file='instances_training.json',
    filter_cfg=dict(filter_empty_gt=True, min_size=32),
    pipeline=pipeline)

train_dataloader = dict(
    batch_size=16,
    num_workers=8,
```

(下页继续)

(续上页)

```
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=True),
    dataset=icdar2015_textdet_train)
```

14.3.5 WildReceiptDataset

该类为 WildReceipt 数据集定制。

标注格式

```
// Close Set
{
  "file_name": "image_files/Image_16/11/d5de7f2a20751e50b84c747c17a24cd98bed3554.jpeg
↪",
  "height": 1200,
  "width": 1600,
  "annotations":
    [
      {
        "box": [550.0, 190.0, 937.0, 190.0, 937.0, 104.0, 550.0, 104.0],
        "text": "SAFEWAY",
        "label": 1
      },
      {
        "box": [1048.0, 211.0, 1074.0, 211.0, 1074.0, 196.0, 1048.0, 196.0],
        "text": "TM",
        "label": 25
      }
    ], //...
}

// Open Set
{
  "file_name": "image_files/Image_12/10/845be0dd6f5b04866a2042abd28d558032ef2576.jpeg
↪",
  "height": 348,
  "width": 348,
  "annotations":
    [
      {
        "box": [114.0, 19.0, 230.0, 19.0, 230.0, 1.0, 114.0, 1.0],
```

(下页继续)

(续上页)

```
        "text": "CHOEUN",
        "label": 2,
        "edge": 1
      },
      {
        "box": [97.0, 35.0, 236.0, 35.0, 236.0, 19.0, 97.0, 19.0],
        "text": "KOREANRESTAURANT",
        "label": 2,
        "edge": 1
      }
    ]
}
```

配置示例

请参考 SDMGR 的配置。

CHAPTER 15

设计理念与特性 [待更新]

待更新

CHAPTER 16

数据流 [待更新]

待更新

CHAPTER 17

模型 [待更新]

待更新

CHAPTER 18

可视化组件 [待更新]

待更新

CHAPTER 19

开发默认约定 [待更新]

待更新

CHAPTER 20

引擎 [待更新]

待更新

CHAPTER 21

支持数据集一览

21.1 支持的数据集

数据集名称	文本检测	文本识别	端到端文本检测识别	关键信息抽取
cocotextv2	✓	✓	✓	
ctw1500	✓	✓	✓	
cute80		✓		
funsd	✓	✓	✓	
icdar2013	✓	✓	✓	
icdar2015	✓	✓	✓	
iiit5k		✓		
mjsynth		✓		
naf	✓	✓	✓	
sroie	✓	✓	✓	
svt	✓	✓	✓	
svtp		✓		

21.2 数据集详情

21.2.1 COCO Text v2

> “COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images”，*arXiv*, 2016. PDF

A. 数据集基础信息

- 官方网址: cocotextv2
- 发布年份: 2016
- 语言: [‘English’]
- 场景: [‘Natural Scene’]
- 标注粒度: [‘Word’]
- 支持任务: [‘textdet’，‘textrecog’，‘textspotting’]
- 数据集许可证: CC BY 4.0

Text Detection/Spotting

```
{
  "cats": {},
  "anns": {
      "45346": {
          "mask":[468.9,286.7,468.9,295.2,493.0,295.8,493.0,287.2],
          "class":"machine printed",
          "bbox":[468.9,286.7,24.1,9.1],
          "image_id":522579,
          "id":167312,
          "language":"english",
          "area":55.5,
          "utf8_string":"the",
          "legibility":"legible"
      },
      // ...
  },
  "imgs": {
      "522579": {
          "file_name":"COCO_train2014_000000522579.jpg",
          "height":476,
          "width":640,
          "id":522579,
          "set":"train",
```

(下页继续)

(续上页)

```
        },
        // ...
    },
    "imgToAnns": {
        "522579": [167294, 167295, 167296, 167297, 167298, 167299, 167300, 167301,␣
↪167302, 167303, 167304, 167305, 167306, 167307, 167308, 167309, 167310, 167311,␣
↪167312, 167313, 167314, 167315, 167316, 167317],
        // ...
    },
    "info": {}
}
```

C. 参考文献

```
@article{veit2016coco, title={Coco-text: Dataset and benchmark for text detection and␣
↪recognition in natural images}, author={Veit, Andreas and Matera, Tomas and Neumann,
↪ Lukas and Matas, Jiri and Belongie, Serge}, journal={arXiv preprint arXiv:1601.
↪07140}, year={2016}}
```

21.2.2 CTW1500

"Curved scene text detection via transverse and longitudinal sequence connection", *PR*, 2019. PDF

A. 数据集基础信息

- 官方网址: ctw1500
- 发布年份: 2019
- 语言: ['English']
- 场景: ['Scene']
- 标注粒度: ['Word', 'Line']
- 支持任务: ['textrecog', 'textdet', 'textspotting']
- 数据集许可证: N/A

C. 参考文献

```
@article{liu2019curved, title={Curved scene text detection via transverse and␣
↪longitudinal sequence connection}, author={Liu, Yuliang and Jin, Lianwen and Zhang,␣
↪Shuaitao and Luo, Canjie and Zhang, Sheng}, journal={Pattern Recognition}, volume=
↪{90}, pages={337--345}, year={2019}, publisher={Elsevier} }
```

21.2.3 CUTE80

"A Robust Arbitrary Text Detection System for Natural Scene Images", *ESWA*, 2014. PDF

A. 数据集基础信息

- 官方网址: cute80
- 发布年份: 2014
- 语言: ['English']
- 场景: ['Natural Scene']
- 标注粒度: ['Word']
- 支持任务: ['textrecog']
- 数据集许可证: N/A

Text Recognition

```
# timage/img_name text 1 text

timage/001.jpg RONALDO 1 RONALDO
timage/002.jpg 7 1 7
timage/003.jpg SEACREST 1 SEACREST
timage/004.jpg BEACH 1 BEACH
```

C. 参考文献

```
@article{risnumawan2014robust, title={A robust arbitrary text detection system for␣
↪natural scene images}, author={Risnumawan, Anhar and Shivakumara, Palaiahankote and␣
↪Chan, Chee Seng and Tan, Chew Lim}, journal={Expert Systems with Applications},␣
↪volume={41}, number={18}, pages={8027--8048}, year={2014}, publisher={Elsevier}}
```

21.2.4 FUNSD

"FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents", *ICDAR*, 2019. PDF

A. 数据集基础信息

- 官方网址: funsd
- 发布年份: 2019
- 语言: ['English']
- 场景: ['Document']
- 标注粒度: ['Word']

- 支持任务: [‘textdet’, ‘textrecog’, ‘textspotting’]
- 数据集许可证: FUNSD License

Text Detection/Recognition/Spotting


```
{
  "form": [
    {
      "id": 0,
      "text": "Registration No.",
      "box": [
          94,
          169,
          191,
          186
      ],
      "linking": [
          [
              0,
              1
          ]
      ],
      "label": "question",
      "words": [
          {
              "text": "Registration",
              "box": [
                  94,
                  169,
                  168,
                  186
              ]
          },
          {
              "text": "No.",
              "box": [
                  170,
                  169,
                  191,
                  183
              ]
          }
      ]
    },
    {
```


(下页继续)

(续上页)


```
      "id": 1,
      "text": "533",
      "box": [
          209,
          169,
          236,
          182
      ],
      "label": "answer",
      "words": [
          {
              "box": [
                  209,
                  169,
                  236,
                  182
              ],
              "text": "533"
          }
      ],
      "linking": [
          [
              0,
              1
          ]
      ]
    }
  ]
}
```


C. 参考文献

```
@inproceedings{jaume2019, title = {FUNSD: A Dataset for Form Understanding in Noisy␣
↪Scanned Documents}, author = {Guillaume Jaume, Hazim Kemal Ekenel, Jean-Philippe␣
↪Thiran}, booktitle = {Accepted to ICDAR-OST}, year = {2019}}
```

21.2.5 Incidental Scene Text IC13

> “ICDAR 2013 Robust Reading Competition”, *ICDAR*, 2013. PDF

A. 数据集基础信息

- 官方网址: icdar2013
- 发布年份: 2013
- 语言: [‘English’]
- 场景: [‘Natural Scene’]
- 标注粒度: [‘Word’]
- 支持任务: [‘textdet’, ‘textrecog’, ‘textspotting’]
- 数据集许可证: N/A

Text Detection

```
# train split
# x1 y1 x2 y2 "transcript"

158 128 411 181 "Footpath"
443 128 501 169 "To"
64 200 363 243 "Colchester"

# test split
# x1, y1, x2, y2, "transcript"

38, 43, 920, 215, "Tiredness"
275, 264, 665, 450, "kills"
0, 699, 77, 830, "A"
```

Text Recognition

```
# img_name, "text"

word_1.png, "PROPER"
word_2.png, "FOOD"
word_3.png, "PRONTO"
```

C. 参考文献

```
@inproceedings{karatzas2013icdar, title={ICDAR 2013 robust reading competition},␣
↪author={Karatzas, Dimosthenis and Shafait, Faisal and Uchida, Seiichi and Iwamura,␣
↪Masakazu and i Bigorda, Lluis Gomez and Mestre, Sergi Robles and Mas, Joan and Mota,
↪ David Fernandez and Almazan, Jon Almazan and De Las Heras, Lluis Pere}, booktitle=
↪{2013 12th international conference on document analysis and recognition}, pages=
↪{1484--1493}, year={2013}, organization={IEEE}}
```

(下页继续)

(续上页)

21.2.6 Incidental Scene Text IC15

> “ICDAR 2015 Competition on Robust Reading”, *ICDAR*, 2015. PDF

A. 数据集基础信息

- 官方网址: icdar2015
- 发布年份: 2015
- 语言: [‘English’]
- 场景: [‘Natural Scene’]
- 标注粒度: [‘Word’]
- 支持任务: [‘textdet’, ‘textrecog’, ‘textspotting’]
- 数据集许可证: CC BY 4.0

Text Detection

```
# x1,y1,x2,y2,x3,y3,x4,y4,trans

377,117,463,117,465,130,378,130,Genaxis Theatre
493,115,519,115,519,131,493,131,[06]
374,155,409,155,409,170,374,170,###
```

Text Recognition

```
# img_name, "text"

word_1.png, "Genaxis Theatre"
word_2.png, "[06]"
word_3.png, "62-03"
```

C. 参考文献

```
@inproceedings{karatzas2015icdar, title={ICDAR 2015 competition on robust reading},
↪author={Karatzas, Dimosthenis and Gomez-Bigorda, Lluis and Nicolaou, Anguelos and
↪Ghosh, Suman and Bagdanov, Andrew and Iwamura, Masakazu and Matas, Jiri and Neumann,
↪ Lukas and Chandrasekhar, Vijay Ramaseshan and Lu, Shijian and others}, booktitle=
↪{2015 13th international conference on document analysis and recognition (ICDAR)},
↪pages={1156--1160}, year={2015}, organization={IEEE}}
```

21.2.7 IIIT5K

> “Scene Text Recognition using Higher Order Language Priors”, *BMVC*, 2012. PDF

A. 数据集基础信息

- 官方网址: iiit5k
- 发布年份: 2012
- 语言: [‘English’]
- 场景: [‘Natural Scene’]
- 标注粒度: [‘Word’]
- 支持任务: [‘textrecog’]
- 数据集许可证: N/A

Text Recognition

```
# img_name, "text"

train/1009_2.png You
train/1017_1.png Rescue
train/1017_2.png mission
```

C. 参考文献

```
@InProceedings{MishraBMVC12, author    = "Mishra, A. and Alahari, K. and Jawahar, C.~
↪V.", title     = "Scene Text Recognition using Higher Order Language Priors",␣
↪booktitle = "BMVC", year      = "2012"}
```

21.2.8 Synthetic Word Dataset (MJSynth/Syn90k)

> “Reading Text in the Wild with Convolutional Neural Networks”, *International Journal of Computer Vision*, 2016. PDF

A. 数据集基础信息

- 官方网址: mjsynth
- 发布年份: 2016
- 语言: [‘English’]
- 场景: [‘Synthesis’]
- 标注粒度: [‘Word’]
- 支持任务: [‘textrecog’]

- 数据集许可证: N/A

Text Recognition

```
./3000/7/182_slinking_71711.jpg 71711
./3000/7/182_REMODELERS_64541.jpg 64541
```

C. 参考文献

```
@InProceedings{Jaderberg14c, author       = "Max Jaderberg and Karen Simonyan and␣
↪Andrea Vedaldi and Andrew Zisserman", title        = "Synthetic Data and Artificial␣
↪Neural Networks for Natural Scene Text Recognition", booktitle    = "Workshop on␣
↪Deep Learning, NIPS", year         = "2014", }
@Article{Jaderberg16, author       = "Max Jaderberg and Karen Simonyan and Andrea␣
↪Vedaldi and Andrew Zisserman", title        = "Reading Text in the Wild with␣
↪Convolutional Neural Networks", journal      = "International Journal of Computer␣
↪Vision", number       = "1", volume       = "116", pages        = "1--20", month    ␣
↪    = "jan", year         = "2016", }
```

21.2.9 NAF

"Deep Visual Template-Free Form Parsing", *ICDAR*, 2019. PDF

A. 数据集基础信息

- 官方网址: naf
- 发布年份: 2019
- 语言: ['English']
- 场景: ['Document', 'Handwritten']
- 标注粒度: ['Word', 'Line']
- 支持任务: ['textrecog', 'textdet', 'textspotting']
- 数据集许可证: CDLA

Text Detection/Recognition/Spotting

```
{"fieldBBs": [{"poly_points": [[435, 1406], [466, 1406], [466, 1439], [435, 1439]],
↪"type": "fieldCheckBox", "id": "f0", "isBlank": 1}, {"poly_points": [[435, 1444],␣
↪[469, 1444], [469, 1478], [435, 1478]], "type": "fieldCheckBox", "id": "f1",
↪"isBlank": 1}],
 "textBBs": [{"poly_points": [[1183, 1337], [2028, 1345], [2032, 1395], [1186, 1398]],
↪ "type": "text", "id": "t0"}, {"poly_points": [[492, 1336], [809, 1338], [809,␣
↪1379], [492, 1378]], "type": "text", "id": "t1"}, {"poly_points": [[512, 1375],␣
↪[798, 1376], [798, 1405], [512, 1404]], "type": "textInst", "id": "t2"}],
↪"imageFilename": "007182398_00026.jpg", "transcriptions": {"f0": "\u00bf\u00bf(下页继续)
↪u00bf \u00bf\u00bf\u00bf 18/1/49 \u00bf\u00bf\u00bf\u00bf\u00bf", "f1": "U.S. Navy␣
↪Const. Batt.", "t0": "APPLICATION FOR HEADSTONE OR MARKER", "t1":
↪"ORIGINAL"}}
```

(续上页)

C. 参考文献

```
@inproceedings{davis2019deep, title={Deep visual template-free form parsing}, author=
↪{Davis, Brian and Morse, Bryan and Cohen, Scott and Price, Brian and Tensmeyer,␣
↪Chris}, booktitle={2019 International Conference on Document Analysis and␣
↪Recognition (ICDAR)}, pages={134--141}, year={2019}, organization={IEEE}}
```

21.2.10 Scanned Receipts OCR and Information Extraction

"ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction", *ICDAR*, 2019. PDF

A. 数据集基础信息

- 官方网址: sroie
- 发布年份: 2019
- 语言: ['English']
- 场景: ['Document']
- 标注粒度: ['Word']
- 支持任务: ['textdet', 'textrecog', 'textspotting']
- 数据集许可证: CC BY 4.0

Text Detection, Text Recognition and Text Spotting

```
# x1,y1,x2,y2,x3,y3,x4,y4,trans

72,25,326,25,326,64,72,64,TAN WOON YANN
50,82,440,82,440,121,50,121,BOOK TA .K(TAMAN DAYA) SDN BND
205,121,285,121,285,139,205,139,789417-W
```

C. 参考文献

```
@INPROCEEDINGS{8977955, author={Huang, Zheng and Chen, Kai and He, Jianhua and Bai,␣
↪Xiang and Karatzas, Dimosthenis and Lu, Shijian and Jawahar, C. V.}, booktitle=
↪{2019 International Conference on Document Analysis and Recognition (ICDAR)}, title=
↪{ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction}, year=
↪{2019}, volume={}, number={}, pages={1516-1520}, doi={10.1109/ICDAR.2019.00244}}
```

21.2.11 Street View Text Dataset (SVT)

"Word Spotting in the Wild", *ECCV*, 2010. PDF

A. 数据集基础信息

- 官方网址: svt
- 发布年份: 2010
- 语言: ['English']
- 场景: ['Natural Scene']
- 标注粒度: ['Word']
- 支持任务: ['textdet', 'textrecog', 'textspotting']
- 数据集许可证: N/A

Text Detection/Recognition/Spotting

```
<image>
  <imageName>img/14_03.jpg</imageName>
  <address>341 Southwest 10th Avenue Portland OR</address>
  <lex>
  LIVING,ROOM,THEATERS,KENNY,ZUKE,DELICATESSEN,CLYDE,COMMON,ACE,HOTEL,PORTLAND,ROSE,
↪CITY,BOOKS,STUMPTOWN,COFFEE,ROASTERS,RED,CAP,GARAGE,FISH,GROTTO,SEAFOOD,RESTAURANT,
↪AURA,RESTAURANT,LOUNGE,ROCCO,PIZZA,PASTA,BUFFALO,EXCHANGE,MARK,SPENCER,LIGHT,FEZ,
↪BALLROOM,READING,FRENZY,ROXY,SCANDALS,MARTINOTTI,CAFE,DELI,CROWSENBERG,HALF
  </lex>
  <Resolution x="1280" y="880"/>
  <taggedRectangles>
    <taggedRectangle height="75" width="236" x="375" y="253">
      <tag>LIVING</tag>
    </taggedRectangle>
    <taggedRectangle height="76" width="175" x="639" y="272">
      <tag>ROOM</tag>
    </taggedRectangle>
    <taggedRectangle height="87" width="281" x="839" y="283">
      <tag>THEATERS</tag>
    </taggedRectangle>
  </taggedRectangles>
</image>
```

C. 参考文献

```
@inproceedings{wang2010word, title={Word spotting in the wild}, author={Wang, Kai and␣
↪Belongie, Serge}, booktitle={European conference on computer vision}, pages={591--
↪604}, year={2010}, organization={Springer}}
```

(下页继续)

(续上页)

21.2.12 Street View Text Perspective (SVT-P)

"Recognizing Text with Perspective Distortion in Natural Scenes", *ICCV*, 2013. PDF

A. 数据集基础信息

- 官方网址: svtp
- 发布年份: 2013
- 语言: ['English']
- 场景: ['Natural Scene']
- 标注粒度: ['Word']
- 支持任务: ['textrecog']
- 数据集许可证: N/A

Text Recognition

```
13_15_0_par.jpg WYNDHAM
13_15_1_par.jpg HOTEL
12_16_0_par.jpg UNITED
```

C. 参考文献

```
@inproceedings{phan2013recognizing, title={Recognizing text with perspective␣
↪distortion in natural scenes}, author={Phan, Trung Quy and Shivakumara,␣
↪Palaiahnakote and Tian, Shangxuan and Tan, Chew Lim}, booktitle={Proceedings of the␣
↪IEEE International Conference on Computer Vision}, pages={569--576}, year={2013}}
```

21.2.13 SynthText in the Wild Dataset

"Synthetic Data for Text Localisation in Natural Images", *CVPR*, 2016. PDF

A. 数据集基础信息

- 官方网址: synthtext
- 发布年份: 2016
- 语言: ['English']
- 场景: ['Synthesis']
- 标注粒度: ['Word', 'Character']

- 支持任务: [‘textdet’, ‘textrecog’, ‘textspotting’]
- 数据集许可证: Synthext Custom

Text Detection/Recognition/Spotting

```
{
    "imnames": [['8/ballet_106_0.jpg', ...]],
    "wordBB": [[[420.58957   418.85016   448.08478   410.3094    117.745026
                 322.30963   322.6857    159.09138   154.27284   260.14597
                 431.9315    427.52274   296.86508    99.56819   108.96211  ]
                [512.3321    431.88342   519.4515    499.81183   179.0544
                 377.97382   376.4993    203.64464   193.77492   313.61514
                 487.58023   484.64633   365.83176   142.49403   144.90457  ]
                [511.92203   428.7077    518.7375    499.0373    172.1684
                 378.35858   377.2078    203.3191    193.0739    319.69186
                 485.6758    482.571     365.76303   142.31898   144.43858  ]
                [420.1795    415.67444   447.3708    409.53485   110.859024
                 322.6944    323.3942    158.76585   153.57182   266.2227
                 430.02707   425.44742   296.79636    99.39314   108.49613  ]]

               [[ 21.06382    46.19922    47.570374   73.95366   197.17792
                    9.993624   48.437763    9.064571   49.659035  208.57095
                  118.41646   162.82489    29.548729    5.800581   28.812992 ]
                [ 23.069519   48.254295   50.130234   77.18146   208.71487
                    8.999153   46.69632     9.698633   50.869553  203.25742
                  122.64043   168.38647    29.660484    6.2558594  29.602367 ]
                [ 41.827087   68.39458    70.03627    98.65903   245.30832
                   30.534437   68.589294   32.57161    73.74529   264.40634
                  147.7303    189.70224    72.08       22.759935   50.81941  ]
                [ 39.82139    66.3395     67.47641    95.43123   233.77136
                   31.528908   70.33074    31.937548   72.534775  269.71988
                  143.50633   184.14066    71.96825    22.304657   50.030033 ]], ...],
    "charBB": [[[423.16126397 439.60847343 450.66887979 466.31976402 479.76190495
                 504.59927448 418.80489444 450.13965942 464.16775197 480.46891089
                 502.46437709 413.02373632 433.01396211 446.7222192  470.28467827
                 482.51674486 116.52285438 139.51408587 150.7448586  162.03366629
                 322.84717946 333.54881536 343.28386485 363.07416389 323.48968759
                 337.98503283 356.66355903 160.48517048 174.1707753  189.64454066
                 155.7637383  167.45490471 179.63644201 262.2183876  271.75848874
                 284.05396524 298.26103738 432.8464733  449.15387392 468.07231897
                 428.11482147 445.61538159 469.24565878 296.86441324 323.6603118
                 344.09880401 101.14677814 110.45423597 120.54555495 131.18342618
                 132.20545124 110.01673682 120.83144568 131.35885673]
                [438.2997574  452.61288403 466.31976402 482.22585715 498.3934528
                 512.20555863 431.88338084 466.11639619 481.73414937 499.62012025
```

(下页继续)

(续上页)

```
  519.36789779 432.51717267 449.23571387 465.73425964 484.45139112
  499.59056304 140.27413679 149.59811175 160.13352083 169.59504507
  333.55849014 344.33923741 361.08275796 378.09844418 339.92898685
  355.57692063 376.51230484 174.1707753  189.07871028 203.64462646
  165.22739457 181.27572412 193.60260894 270.99557614 283.13281739
  298.75499435 313.61511672 447.1421735  470.27065563 487.02126631
  446.97485257 468.98979567 484.64633864 317.88691577 341.16094163
  365.8300006  111.15280603 120.54555495 130.72086821 135.27663717
  142.4726875  120.1331955  133.07976304 144.75919258]
 [435.54895424 449.95797159 464.5848793  480.68235876 497.04793842
  511.1101386  428.95660757 463.61882066 480.14247127 498.2535215
  518.03243928 429.36600266 447.19056345 463.89483785 482.21016814
  498.18529977 142.63162835 152.55587851 162.80539142 172.21885945
  333.35620309 344.09880401 360.86201193 377.82379299 339.7646859
  355.37508239 376.1110999  172.46032372 187.37816388 201.39094518
  163.04321987 178.99078221 191.89681939 275.3073355  286.08373072
  301.85539131 318.57227103 444.54207279 467.53925436 485.27070558
  444.57367155 466.90671029 482.56302723 317.62908407 340.9131681
  365.44465854 109.40501176 119.4999228  129.67892444 134.35253232
  140.97421069 118.61779828 131.34019115 143.25688164]
 [420.17946701 436.74150236 448.74896556 464.5848793  478.18853922
  503.4152019  415.67442461 447.3707845  462.35927516 478.8614766
  500.86810735 409.54560397 430.77026495 444.64606264 467.79077782
  480.89051912 119.14629674 142.63162835 153.56593297 164.78799774
  322.69436747 333.35620309 343.11884239 362.84714115 323.37931952
  337.83763574 356.35573621 158.76583616 172.46032372 187.37816388
  153.57183805 165.15781218 177.92125239 266.22269514 274.45156305
  286.82608962 302.69695881 430.02705241 446.01814255 466.05208347
  425.44741792 443.19481667 466.90671029 296.79634428 323.49707084
  343.82488703  99.39315359 109.40501176 119.4999228  130.25798537
  130.70149005 108.49612777 119.08444238 129.84935461]]

[[ 22.26958901  21.60559248  27.0241972   27.25747678  27.45783459
   28.73896576  47.91255579  47.80732383  53.77711568  54.24219042
   52.00169325  74.79043429  80.45929285  81.04748707  76.11658669
   82.58335942 203.67278213 201.2743445  205.59358622 205.51198143
   10.06536976  10.82312635  16.77203865  16.31842372  54.80444433
   54.66492     47.33822371  15.08534083  15.18716407   9.62607092
   51.06813224  50.18928243  56.16019366 220.78902143 236.08062638
  231.69267533 209.73652786 124.25352842 119.99631725 128.73732717
  165.78411123 167.31764153 167.05531699  29.97351822  31.5116502
   31.14650552   5.88513488  12.51324147  12.57920537   8.21515307
    8.21998849  35.66412031  29.17945741  36.00660903]
```

(下页继续)

(续上页)

```
               [ 22.46075572  21.76391911  27.25747678  27.49456029  27.73554156
                 28.85582217  48.25428361  48.21714995  54.27828788  54.78857757
                 52.4595556   75.57743634  81.15533616  81.86325615  76.681392
                 83.31596322 210.04771309 203.83983042 208.00417391 207.41791524
                  9.79265706  10.55231862  16.36406888  15.97405105  54.64620856
                 54.49559004  47.09756263  15.18716407  15.29808166   9.69862498
                 51.27597632  50.48652154  56.49239954 216.92183074 232.02141018
                226.44624213 203.25738931 125.19349641 121.32658508 130.00428964
                167.43676857 169.36588297 168.38645076  29.58279603  31.19899202
                 30.75826599   5.92344996  12.57920537  12.64571832   8.23451892
                  8.26856497  35.82646468  29.342662    36.22165159]
               [ 40.15739982  40.47241401  40.79219178  41.14411963  41.50190876
                 41.80934074  66.81590976  68.05921213  68.6519006   69.30152766
                 70.01097963  96.14641662  96.04484417  96.89110144  97.81897661
                 98.62829468 237.26055111 240.35280825 243.54641271 245.04022528
                 31.33842788  31.14650552  30.84702178  30.54399042  69.80098672
                 68.7212013   68.62479627  32.13243303  32.34474067  32.54416771
                 72.82501686  73.31372392  73.70922459 267.74318222 265.39839711
                259.52741156 253.14023308 144.60810334 145.23371653 147.69958337
                186.00278322 188.17713786 189.70144388  71.89351759  53.62266986
                 54.40060855  22.41084398  22.51791234  22.62587258  17.11356079
                 22.74567232  50.25232032  46.05692507  50.79345235]
               [ 39.82138755  40.18347166  40.44598236  40.79219178  41.08959901
                 41.64111176  66.33948982  67.47640971  68.01403337  68.60595247
                 69.3953105   95.13188979  95.21297344  95.91593691  97.08847413
                 97.75212171 229.94285119 237.26055111 240.66752705 242.74145162
                 31.52890731  31.33842788  31.16401306  30.81155638  69.87135926
                 68.80273568  68.71664209  31.93753588  32.13243303  32.34474067
                 72.53476992  72.88981775  73.28094858 269.71986636 267.92938572
                262.93698624 256.88902439 143.50635029 143.61251781 146.24080653
                184.14064261 185.86853729 188.17713786  71.96823746  53.79651809
                 54.60870874  22.30465649  22.41084398  22.51791234  17.07939535
                 22.63671808  50.03002471  45.81009198  50.49899163]], ...],
    "txt": [['Lines:\nI lost\nKevin ' 'will                ' 'line\nand           '
             'and\nthe             ' '(and                ' 'the\nout            '
             'you                 ' "don't\n pkg          "], ...]
}
```

C. 参考文献

```
@InProceedings{Gupta16, author       = "Ankush Gupta and Andrea Vedaldi and Andrew␣
↪Zisserman", title        = "Synthetic Data for Text Localisation in Natural Images",
↪ booktitle    = "IEEE Conference on Computer Vision and Pattern Recognition", year ␣
↪       = "2016", }
```

21.2.14 Text OCR

"TextOCR: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text", *CVPR*, 2021. PDF

A. 数据集基础信息

- 官方网址: textocr
- 发布年份: 2021
- 语言: ['English']
- 场景: ['Natural Scene']
- 标注粒度: ['Word']
- 支持任务: ['textdet', 'textrecog', 'textspotting']
- 数据集许可证: CC BY 4.0

Text Detection/Recognition/Spotting

```
{
  "imgs": {
    "OpenImages_ImageID_1": {
      "id": "OpenImages_ImageID_1",
      "width": "INT, Width of the image",
      "height": "INT, Height of the image",
      "set": "Split train|val|test",
      "filename": "train|test/OpenImages_ImageID_1.jpg"
    },
    "OpenImages_ImageID_2": {
      "...": "..."
    }
  },
  "anns": {
    "OpenImages_ImageID_1_1": {
      "id": "STR, OpenImages_ImageID_1_1, Specifies the nth annotation for an image",
      "image_id": "OpenImages_ImageID_1",
      "bbox": [
        "FLOAT x1",
        "FLOAT y1",
        "FLOAT x2",
        "FLOAT y2"
      ],
      "points": [
        "FLOAT x1",
        "FLOAT y1",
        "FLOAT x2",
```

(下页继续)

(续上页)

```
        "FLOAT y2",
        "...",
        "FLOAT xN",
        "FLOAT yN"
      ],
      "utf8_string": "text for this annotation",
      "area": "FLOAT, area of this box"
    },
    "OpenImages_ImageID_1_2": {
      "...": "..."
    },
    "OpenImages_ImageID_2_1": {
      "...": "..."
    }
  },
  "img2Anns": {
    "OpenImages_ImageID_1": [
      "OpenImages_ImageID_1_1",
      "OpenImages_ImageID_1_2",
      "OpenImages_ImageID_1_2"
    ],
    "OpenImages_ImageID_N": [
      "..."
    ]
  }
}
```

C. 参考文献

```
@inproceedings{singh2021textocr, title={{TextOCR}: Towards large-scale end-to-end␣
↪reasoning for arbitrary-shaped scene text}, author={Singh, Amanpreet and Pang, Guan␣
↪and Toh, Mandy and Huang, Jing and Galuba, Wojciech and Hassner, Tal}, journal={The␣
↪Conference on Computer Vision and Pattern Recognition}, year={2021}}
```

21.2.15 Total Text

"Total-Text: Towards Orientation Robustness in Scene Text Detection", *IJDAR*, 2020. PDF

A. 数据集基础信息

- 官方网址: totaltext
- 发布年份: 2020
- 语言: ['English']

- 场景: [‘Natural Scene’]
- 标注粒度: [‘Word’]
- 支持任务: [‘textdet’, ‘textrecog’, ‘textspotting’]
- 数据集许可证: BSD-3

Text Detection/Spotting

```
x: [[259 313 389 427 354 302]], y: [[542 462 417 459 507 582]], ornt: [u'c'],␣
↪transcriptions: [u'PAUL']
x: [[400 478 494 436]], y: [[398 380 448 465]], ornt: [u'#'], transcriptions: [u'#']
```

C. 参考文献

```
@article{CK2019, author = {Chee Kheng Chng and Chee Seng Chan and Chenglin Liu},␣
↪title = {Total-Text: Towards Orientation Robustness in Scene Text Detection},␣
↪journal = {International Journal on Document Analysis and Recognition (IJDAR)},␣
↪volume = {23}, pages = {31-52}, year = {2020}, doi = {10.1007/s10032-019-00334-z}}
```

21.2.16 WildReceipt

“Spatial Dual-Modality Graph Reasoning for Key Information Extraction”, *arXiv*, 2021. PDF

A. 数据集基础信息

- 官方网址: wildreceipt
- 发布年份: 2021
- 语言: [‘English’]
- 场景: [‘Receipt’]
- 标注粒度: [‘Word’]
- 支持任务: [‘kie’, ‘textdet’, ‘textrecog’, ‘textspotting’]
- 数据集许可证: N/A

KIE

```
// Close Set
{
  "file_name": "image_files/Image_16/11/d5de7f2a20751e50b84c747c17a24cd98bed3554.jpeg
↪",
  "height": 1200,
  "width": 1600,
  "annotations":
```

(下页继续)

(续上页)

```
    [
      {
        "box": [550.0, 190.0, 937.0, 190.0, 937.0, 104.0, 550.0, 104.0],
        "text": "SAFEWAY",
        "label": 1
      },
      {
        "box": [1048.0, 211.0, 1074.0, 211.0, 1074.0, 196.0, 1048.0, 196.0],
        "text": "TM",
        "label": 25
      }
    ], //...
}

// Open Set
{
  "file_name": "image_files/Image_12/10/845be0dd6f5b04866a2042abd28d558032ef2576.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": 2,
        "edge": 1
      },
      {
        "box": [97.0, 35.0, 236.0, 35.0, 236.0, 19.0, 97.0, 19.0],
        "text": "KOREANRESTAURANT",
        "label": 2,
        "edge": 1
      }
    ]
}
```

C. 参考文献

```
@article{sun2021spatial, title={Spatial Dual-Modality Graph Reasoning for Key␣
↪Information Extraction}, author={Sun, Hongbin and Kuang, Zhanghui and Yue, Xiaoyu␣
↪and Lin, Chenhao and Zhang, Wayne}, journal={arXiv preprint arXiv:2103.14470}, year=
↪{2021} }
```

CHAPTER 22

数据准备 (Beta)

注解： Dataset Preparer 目前仍处在公测阶段，欢迎尝鲜试用！如遇到任何问题，请及时向我们反馈。

22.1 一键式数据准备脚本

MMOCR 提供了统一的一站式数据集准备脚本 `prepare_dataset.py`。

仅需一行命令即可完成数据的下载、解压、格式转换，及基础配置的生成。

```
python tools/dataset_converters/prepare_dataset.py [-h] [--nproc NPROC] [--task
↪{textdet,textrecog,textspotting,kie}] [--splits SPLITS [SPLITS ...]] [--lmdb] [--
↪overwrite-cfg] [--dataset-zoo-path DATASET_ZOO_PATH] datasets [datasets ...]
```

例如，以下命令展示了如何使用该脚本为 ICDAR2015 数据集准备文本检测任务所需的数据。

```
python tools/dataset_converters/prepare_dataset.py icdar2015 --task textdet --
↪overwrite-cfg
```

该脚本也支持同时准备多个数据集，例如，以下命令展示了如何使用该脚本同时为 ICDAR2015 和 TotalText 数据集准备文本识别任务所需的数据。

```
python tools/dataset_converters/prepare_dataset.py icdar2015 totaltext --task␣
↪textrecog --overwrite-cfg
```

进一步了解 Dataset Preparer 支持的数据集，您可以浏览支持的数据集文档。一些需要手动准备的数据集也列在了文字检测 和文字识别 内。

对于中国境内的用户，我们也推荐通过开源数据平台OpenDataLab来下载数据，以获得更好的下载体验。数据下载后，参考脚本中 data_obtainer 的 save_name 字段，将文件放在 data/cache/ 下并重新运行脚本即可。

22.2 进阶用法

22.2.1 LMDB 格式

在文本识别任务中，通常使用 LMDB 格式来存储数据，以加快数据的读取速度。在使用 prepare_dataset.py 脚本准备数据时，可以通过 --lmdb 参数来指定将数据转换为 LMDB 格式。例如：

```
python tools/dataset_converters/prepare_dataset.py icdar2015 --task textrecog --lmdb
```

数据集准备完成后，Dataset Preparer 会在 configs/textrecog/_base_/datasets/ 中生成 icdar2015_lmdb.py 配置。你可以继承该配置，并将 dataloader 指向 LMDB 数据集。然而，LMDB 数据集的读取需要配合 LoadImageFromNDArray，因此你也同样需要修改 pipeline。

例如，想要将 configs/textrecog/crnn/crnn_mini-vgg_5e_mj.py 的训练集改为刚刚生成的 icdar2015，则需要作如下修改：

1. 修改 configs/textrecog/crnn/crnn_mini-vgg_5e_mj.py:

```
_base_ = [
    '../_base_/datasets/icdar2015_lmdb.py',  # 指向 icdar2015 lmdb 数据集
     ... # 省略
]

train_list = [_base_.icdar2015_lmdb_textrecog_train]
...
```

2. 修改 configs/textrecog/crnn/_base_crnn_mini-vgg.py 中的 train_pipeline，将 LoadImageFromFile 改为 LoadImageFromNDArray：

```
train_pipeline = [
 dict(
     type='LoadImageFromNDArray',
     color_type='grayscale',
     file_client_args=file_client_args,
     ignore_empty=True,
     min_size=2),
```

(下页继续)

(续上页)

```
  ...
]
```

22.3 设计

OCR 数据集数量众多，不同的数据集有着不同的语言，不同的标注格式，不同的场景等。数据集的使用情况一般有两种，一种是快速的了解数据集的相关信息，另一种是在使用数据集训练模型。为了满足这两种使用场景 MMOCR 提供数据集自动化准备脚本，数据集自动化准备脚本使用了模块化的设计，极大地增强了扩展性，用户能够很方便地配置其他公开数据集或私有数据集。数据集自动化准备脚本的配置文件被统一存储在 `dataset_zoo/` 目录下，用户可以在该目录下找到所有已由 MMOCR 官方支持的数据集准备脚本配置文件。该文件夹的目录结构如下：

```
dataset_zoo/
├── icdar2015
│   ├── metafile.yml
│   ├── sample_anno.md
│   ├── textdet.py
│   ├── textrecog.py
│   └── textspotting.py
└── wildreceipt
    ├── metafile.yml
    ├── sample_anno.md
    ├── kie.py
    ├── textdet.py
    ├── textrecog.py
    └── textspotting.py
```

22.3.1 数据集相关信息

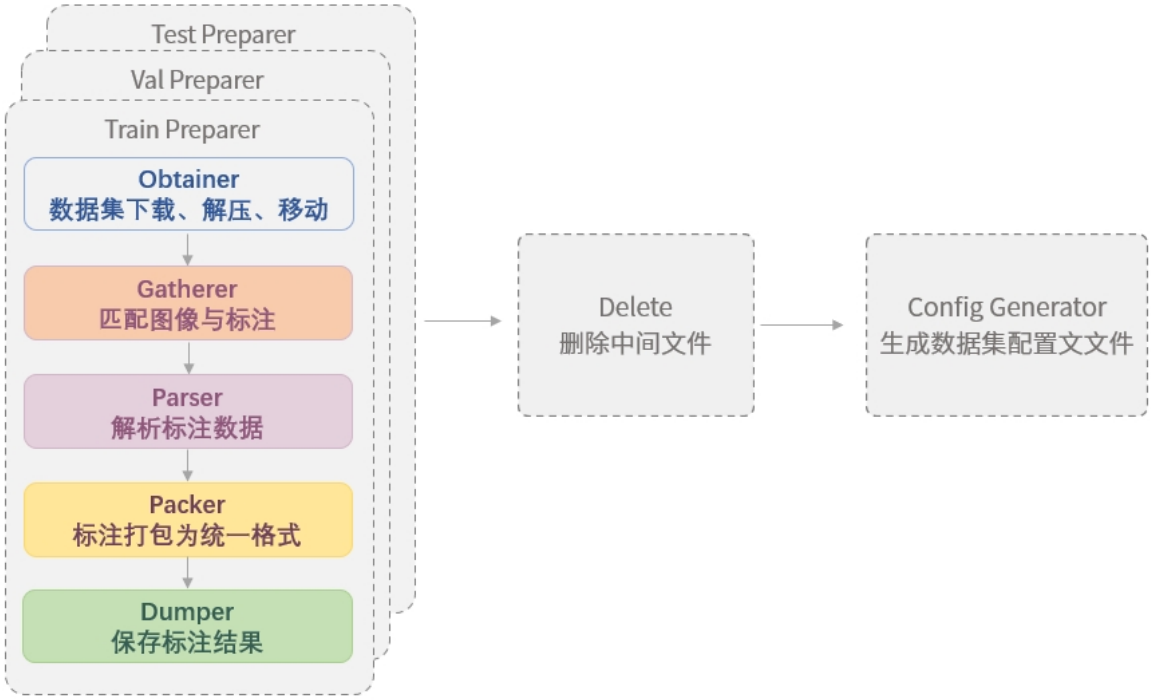
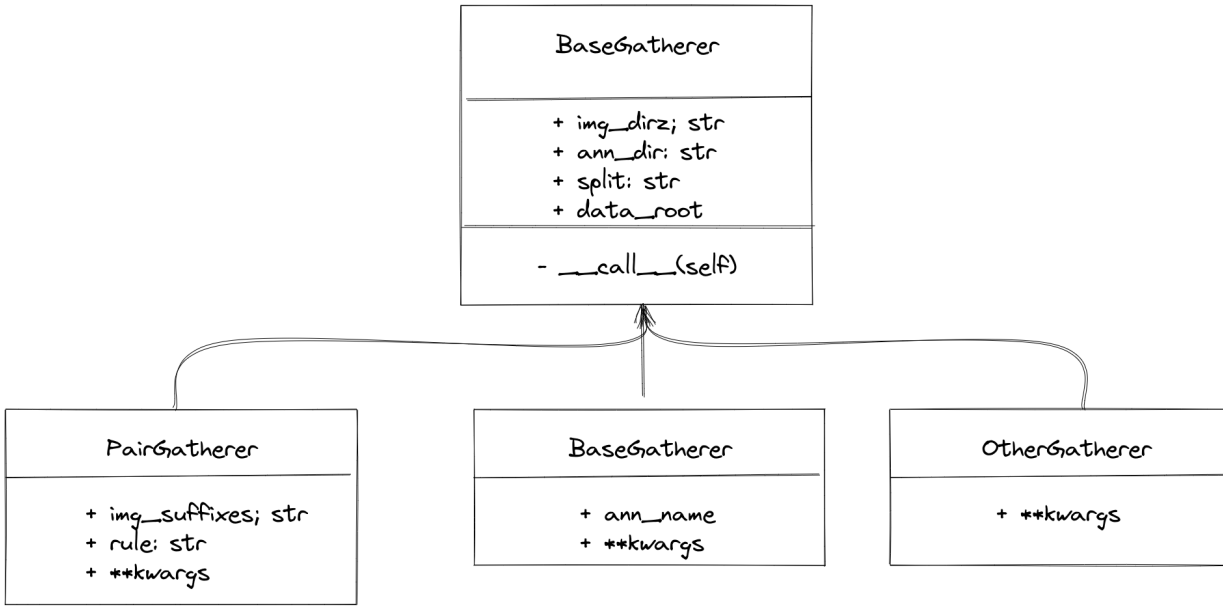
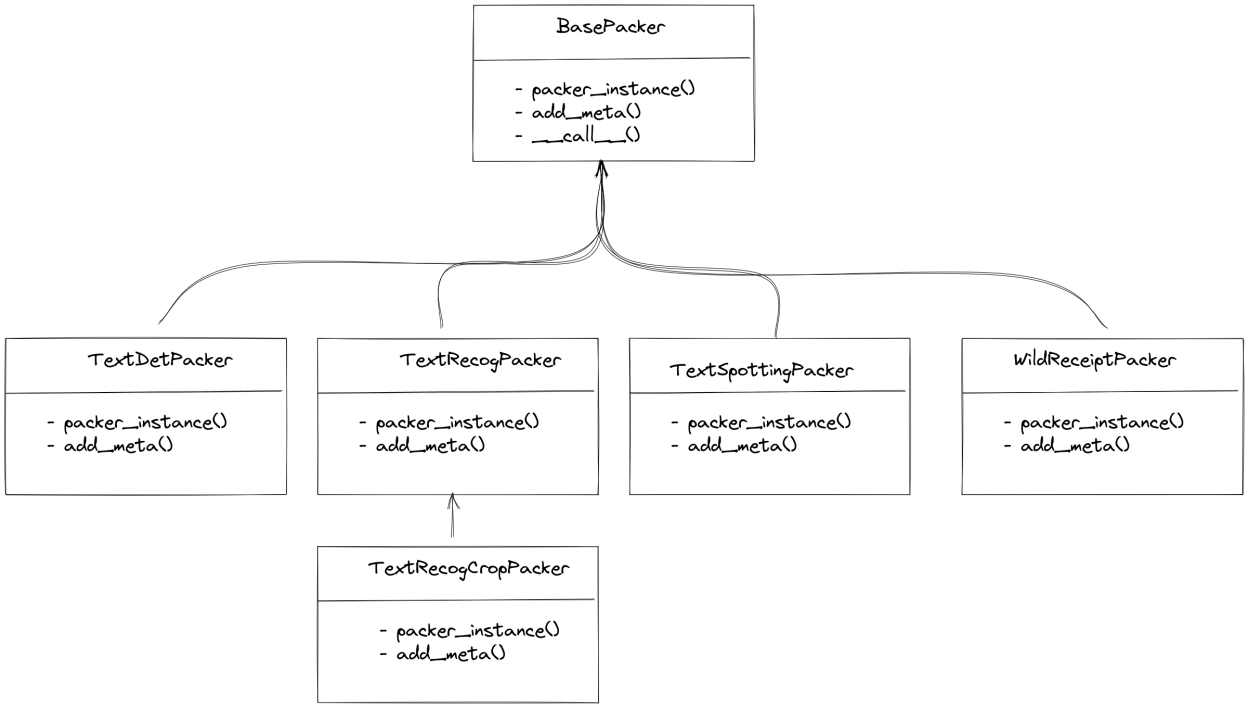
数据集的相关信息包括数据集的标注格式、数据集的标注示例、数据集的基本统计信息等。虽然在每个数据集的官网中都有这些信息，但是这些信息分散在各个数据集的官网中，用户需要花费大量的时间来挖掘数据集的基本信息。因此，MMOCR 设计了一些范式，它可以帮助用户快速了解数据集的基本信息。MMOCR 将数据集的相关信息分为两个部分，一部分是数据集的基本信息包括包括发布年份，论文作者，以及版权等其他信息，另一部分是数据集的标注信息，包括数据集的标注格式、数据集的标注示例。每一部分 MMOCR 都会提供一个范式，贡献者可以根据范式来填写数据集的基本信息，使用用户就可以快速了解数据集的基本信息。根据数据集的基本信息 MMOCR 提供了一个 `metafile.yml` 文件，其中存放了对应数据集的基本信息，包括发布年份，论文作者，以及版权等其他信息，这样用户就可以快速了解数据集的基本信息。该文件在数据集准备过程中并不是强制要求的（因此用户在使用添加自己的私有数据集时可以忽略该文件），但为了用户更好地了解各个公开数据集的信息，MMOCR 建议用户在使用数据集准备脚本前阅读对应的元文件信息，以了解该数据集的特征是否符合用户需求。MMOCR 以 ICDAR2015 作为示例，其示例内容如下所示：

```
Name: 'Incidental Scene Text IC15'
Paper:
  Title: ICDAR 2015 Competition on Robust Reading
  URL: https://rrc.cvc.uab.es/files/short_rrc_2015.pdf
  Venue: ICDAR
  Year: '2015'
  BibTeX: '@inproceedings{karatzas2015icdar,
  title={ICDAR 2015 competition on robust reading},
  author={Karatzas, Dimosthenis and Gomez-Bigorda, Lluis and Nicolaou, Anguelos and␣
↪Ghosh, Suman and Bagdanov, Andrew and Iwamura, Masakazu and Matas, Jiri and Neumann,
↪ Lukas and Chandrasekhar, Vijay Ramaseshan and Lu, Shijian and others},
  booktitle={2015 13th international conference on document analysis and recognition␣
↪(ICDAR)},
  pages={1156--1160},
  year={2015},
  organization={IEEE}}'
Data:
  Website: https://rrc.cvc.uab.es/?ch=4
  Language:
    - English
  Scene:
    - Natural Scene
  Granularity:
    - Word
  Tasks:
    - textdet
    - textrecog
    - textspotting
  License:
    Type: CC BY 4.0
    Link: https://creativecommons.org/licenses/by/4.0/
```

具体地，MMOCR 在下表中列出每个字段对应的含义：

对于数据集的标注信息，MMOCR 提供了一个 `sample_anno.md` 文件，用户可以根据范式来填写数据集的标注信息，这样用户就可以快速了解数据集的标注信息。MMOCR 以 ICDAR2015 作为示例，其示例内容如下所示：

````
    **Text Detection**

    ```text
    # x1,y1,x2,y2,x3,y3,x4,y4,trans

    377,117,463,117,465,130,378,130,Genaxis Theatre
````

(下页继续)
````

(续上页)

```
    493,115,519,115,519,131,493,131,[06]
    374,155,409,155,409,170,374,170,###
    ```
```

`sample_anno.md` 中包含数据集针对不同任务的标注信息，包含标注文件的格式 (text 对应的是 txt 文件，标注文件的格式也可以在 meta.yml 中找到)，标注的示例。

通过上述两个文件的信息，用户就可以快速了解数据集的基本信息，同时 MMOCR 汇总了所有数据集的基本信息，用户可以在 Overview 中查看所有数据集的基本信息。

### 22.3.2 数据集使用

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。因此，为了方便用户进行数据集准备，我们设计了 Dataset Preaprer，帮助用户快速将数据集准备为 MMOCR 支持的格式, 详见数据格式文档。下图展示了 Dataset Preparer 的典型运行流程。




由图可见，Dataset Preparer 在运行时，会依次执行以下操作：

1. 对训练集、验证集和测试集，由各 preparer 进行：
    1. 数据集的下载、解压、移动（*Obtainer*）
    2. 匹配标注与图像（*Gatherer*）
    3. 解析原标注（*Parser*）
```

4. 打包标注为统一格式（*Packer*）
5. 保存标注（*Dumper*）

2. 删除文件（Delete）
3. 生成数据集的配置文件（Config Generator）

为了便于应对各种数据集的情况，MMOCR 将每个部分均设计为可插拔的模块，并允许用户通过 dataset_zoo/ 下的配置文件对数据集准备流程进行配置。这些配置文件采用了 Python 格式，其使用方法与 MMOCR 算法库的其他配置文件完全一致，详见配置文件文档。

在 `dataset_zoo/` 下，每个数据集均占有一个文件夹，文件夹下会以任务名命名配置文件，以区分不同任务下的配置。以 ICDAR2015 文字检测部分为例，示例配置 `dataset_zoo/icdar2015/textdet.py` 如下所示：

```
data_root = 'data/icdar2015'
cache_path = 'data/cache'
train_preparer = dict(
    obtainer=dict(
        type='NaiveDataObtainer',
        cache_path=cache_path,
        files=[
            dict(
                url='https://rrc.cvc.uab.es/downloads/ch4_training_images.zip',
                save_name='ic15_textdet_train_img.zip',
                md5='c51cbace155dcc4d98c8dd19d378f30d',
                content=['image'],
                mapping=[['ic15_textdet_train_img', 'textdet_imgs/train']]),
            dict(
                url='https://rrc.cvc.uab.es/downloads/'
                'ch4_training_localization_transcription_gt.zip',
                save_name='ic15_textdet_train_gt.zip',
                md5='3bfaf1988960909014f7987d2343060b',
                content=['annotation'],
                mapping=[['ic15_textdet_train_gt', 'annotations/train']]),
        ]),
    gatherer=dict(
        type='PairGatherer',
        img_suffixes=['.jpg', '.JPG'],
        rule=[r'img_(\d+)\.([jJ][pP][gG])', r'gt_img_\1.txt']),
    parser=dict(type='ICDARTxtTextDetAnnParser', encoding='utf-8-sig'),
    packer=dict(type='TextDetPacker'),
    dumper=dict(type='JsonDumper'),
)

test_preparer = dict(
```

(下页继续)

(续上页)

```
    obtainer=dict(
        type='NaiveDataObtainer',
        cache_path=cache_path,
        files=[
            dict(
                url='https://rrc.cvc.uab.es/downloads/ch4_test_images.zip',
                save_name='ic15_textdet_test_img.zip',
                md5='97e4c1ddcf074ffcc75feff2b63c35dd',
                content=['image'],
                mapping=[['ic15_textdet_test_img', 'textdet_imgs/test']]),
            dict(
                url='https://rrc.cvc.uab.es/downloads/'
                'Challenge4_Test_Task4_GT.zip',
                save_name='ic15_textdet_test_gt.zip',
                md5='8bce173b06d164b98c357b0eb96ef430',
                content=['annotation'],
                mapping=[['ic15_textdet_test_gt', 'annotations/test']]),
        ]),
    gatherer=dict(
        type='PairGatherer',
        img_suffixes=['.jpg', '.JPG'],
        rule=[r'img_(\d+)\.([jJ][pP][gG])', r'gt_img_\1.txt']),
    parser=dict(type='ICDARTxtTextDetAnnParser', encoding='utf-8-sig'),
    packer=dict(type='TextDetPacker'),
    dumper=dict(type='JsonDumper'),
)

delete = ['annotations', 'ic15_textdet_test_img', 'ic15_textdet_train_img']
config_generator = dict(type='TextDetConfigGenerator')
```

数据集下载、解压、移动 (Obtainer)

Dataset Preparer 中，`obtainer` 模块负责了数据集的下载、解压和移动。如今，MMOCR 暂时只提供了 `NaiveDataObtainer`。通常来说，内置的 `NaiveDataObtainer` 即可完成绝大部分可以通过直链访问的数据集的下载，并支持解压、移动文件和重命名等操作。然而，MMOCR 暂时不支持自动下载存储在百度或谷歌网盘等需要登陆才能访问资源的数据集。这里简要介绍一下 `NaiveDataObtainer`.

`files` 字段是一个列表，列表中的每个元素都是一个字典，用于描述一个数据集文件的下载信息。如下表所示：

同时，Dataset Preparer 存在以下约定：

- 不同类型的数据集的图片统一移动到对应类别 `{taskname}_imgs/{split}/`文件夹下，如 `textdet_imgs/train/`。

- 对于一个标注文件包含所有图像的标注信息的情况，标注移到到 `annotations/{split}.*` 文件中。如 `annotations/train.json`。
- 对于一个标注文件包含一个图像的标注信息的情况，所有的标注文件移动到 `annotations/{split}/`文件中。如 `annotations/train/`。
- 对于一些其他的特殊情况，比如所有训练、测试、验证的图像都在一个文件夹下，可以将图像移动到自己设定的文件夹下，比如 `{taskname}_imgs/imgs/`，同时要在后续的 `gatherer` 模块中指定图像的存储位置。

示例配置如下：

```
    obtainer=dict(
        type='NaiveDataObtainer',
        cache_path=cache_path,
        files=[
            dict(
                url='https://rrc.cvc.uab.es/downloads/ch4_training_images.zip',
                save_name='ic15_textdet_train_img.zip',
                md5='c51cbace155dcc4d98c8dd19d378f30d',
                content=['image'],
                mapping=[['ic15_textdet_train_img', 'textdet_imgs/train']]),
            dict(
                url='https://rrc.cvc.uab.es/downloads/'
                'ch4_training_localization_transcription_gt.zip',
                save_name='ic15_textdet_train_gt.zip',
                md5='3bfaf1988960909014f7987d2343060b',
                content=['annotation'],
                mapping=[['ic15_textdet_train_gt', 'annotations/train']]),
        ]),
```

数据集收集 (Gatherer)

`gatherer` 遍历数据集目录下的文件，将图像与标注文件一一对应，并整理出一份文件列表供 `parser` 读取。因此，首先需要知道当前数据集下，图片文件与标注文件匹配的规则。OCR 数据集有两种常用标注保存形式，一种为多个标注文件对应多张图片，一种则为单个标注文件对应多张图片，如：

```
多对多
├── {taskname}_imgs/{split}/img_img_1.jpg
├── annotations/{split}/gt_img_1.txt
├── {taskname}_imgs/{split}/img_2.jpg
├── annotations/{split}/gt_img_2.txt
├── {taskname}_imgs/{split}/img_3.JPG
├── annotations/{split}/gt_img_3.txt
```

(下页继续)

(续上页)

```
单对多
├── {taskname}/{split}/img_1.jpg
├── {taskname}/{split}/img_2.jpg
├── {taskname}/{split}/img_3.JPG
├── annotations/gt.txt
```



具体设计如下所示

MMOCR 内置了 `PairGatherer` 与 `MonoGatherer` 来处理以上这两种常用情况。其中 `PairGatherer` 用于多对多的情况，`MonoGatherer` 用于单对多的情况。

注解： 为了简化处理，gatherer 约定数据集的图片和标注需要分别储存在 `{taskname}_imgs/{split}/` 和 `annotations/` 下。特别地，对于多对多的情况，标注文件需要放置于 `annotations/{split}`。

- 在多对多的情况下，`PairGatherer` 需要按照一定的命名规则找到图片文件和对应的标注文件。首先，需要通过 `img_suffixes` 参数指定图片的后缀名，如上述例子中的 `img_suffixes=[.jpg,.JPG]`。此外，还需要通过正则表达式 `rule`，来指定图片与标注文件的对应关系，其中，规则 `rule` 是一个**正则表达式对**，例如 `rule=[r'img_(\d+)\.([jJ][pP][gG])', r'gt_img_\1.txt']`。第一个正则表达式用于匹配图片文件名，`\d+` 用于匹配图片的序号，`([jJ][pP][gG])` 用于匹配图片的后缀名。第二个正则表达式用于匹配标注文件名，其中 `\1` 则将匹配到的图片序号与标注文件序号对应起来。示例配置为

```
gatherer=dict(
    type='PairGatherer',
    img_suffixes=['.jpg', '.JPG'],
    rule=[r'img_(\d+)\.([jJ][pP][gG])', r'gt_img_\1.txt']),
```

- 单对多的情况通常比较简单，用户只需要指定标注文件名即可。对于训练集示例配置为

```
    gatherer=dict(type='MonoGatherer', ann_name='train.txt'),
```

MMOCR 同样对 Gatherer 的返回值做了约定，Gatherer 会返回两个元素的元组，第一个元素为图像路径列表 (包含所有图像路径) 或者所有图像所在的文件夹，第二个元素为标注文件路径列表 (包含所有标注文件路径) 或者标注文件的路径 (该标注文件包含所有图像标注信息)。具体而言，PairGatherer 的返回值为 (图像路径列表，标注文件路径列表)，示例如下：

```
    (['{taskname}_imgs/{split}/img_1.jpg', '{taskname}_imgs/{split}/img_2.jpg', '
↪{taskname}_imgs/{split}/img_3.JPG'],
    ['annotations/{split}/gt_img_1.txt', 'annotations/{split}/gt_img_2.txt',
↪'annotations/{split}/gt_img_3.txt'])
```

MonoGatherer 的返回值为 (图像文件夹路径，标注文件路径)，示例为：

```
    ('{taskname}/{split}', 'annotations/gt.txt')
```

数据集解析 (Parser)

Parser 主要用于解析原始的标注文件，因为原始标注情况多种多样，因此 MMOCR 提供了 BaseParser 作为基类，用户可以继承该类来实现自己的 Parser。在 BaseParser 中，MMOCR 设计了两个接口：parse_files 和 parse_file，约定在其中进行标注的解析。而对于 Gatherer 的两种不同输入情况（多对多、单对多），这两个接口的实现则应有所不同。

- BaseParser 默认处理**多对多**的情况。其中，由 parer_files 将数据并行分发至多个 parse_file 进程，并由每个 parse_file 分别进行单个图像标注的解析。
- 对于**单对多**的情况，用户则需要重写 parse_files，以实现加载标注，并返回规范的结果。

BaseParser 的接口定义如下所示：

```
class BaseParser:

    def __call__(self, img_paths, ann_paths):
        return self.parse_files(img_paths, ann_paths)

    def parse_files(self, img_paths: Union[List[str], str],
                    ann_paths: Union[List[str], str]) -> List[Tuple]:
        samples = track_parallel_progress_multi_args(
            self.parse_file, (img_paths, ann_paths), nproc=self.nproc)
        return samples

    @abstractmethod
    def parse_file(self, img_path: str, ann_path: str) -> Tuple:

        raise NotImplementedError
```

为了保证后续模块的统一性，MMOCR 对 `parse_files` 与 `parse_file` 的返回值做了约定。`parse_file` 的返回值为一个元组，元组中的第一个元素为图像路径，第二个元素为标注信息。标注信息为一个列表，列表中的每个元素为一个字典，字典中的字段为 `poly`, `text`, `ignore`，如下所示：

```
# An example of returned values:
(
    'imgs/train/xxx.jpg',
    [
        dict(
            poly=[0, 1, 1, 1, 1, 0, 0, 0],
            text='hello',
            ignore=False),
        ...
    ]
)
```

`parse_files` 的输出为一个列表，列表中的每个元素为 `parse_file` 的返回值。示例为：

```
[
    (
        'imgs/train/xxx.jpg',
        [
            dict(
                poly=[0, 1, 1, 1, 1, 0, 0, 0],
                text='hello',
                ignore=False),
            ...
        ]
    ),
    ...
]
```

数据集转换 (Packer)

`packer` 主要是将数据转化到统一的标注格式，因为输入的数据为 Parsers 的输出，格式已经固定，因此 Packer 只需要将输入的格式转化为每种任务统一的标注格式即可。如今 MMOCR 支持的任务有文本检测、文本识别、端对端 OCR 以及关键信息提取，MMOCR 针对每个任务均有对应的 Packer，如下所示：



对于文字检测、端对端 OCR 及关键信息提取，MMOCR 均有唯一对应的 `Packer`。而在文字识别领域，MMOCR 则提供了两种 `Packer`，分别为 `TextRecogPacker` 和 `TextRecogCropPacker`，其原因在与文字识别的数据集存在两种情况：

- 每个图像均为一个识别样本，`parser` 返回的标注信息仅为一个 `dict(text='xxx')`，此时使用 `TextRecogPacker` 即可。
- 数据集没有将文字从图像中裁剪出来，本质是一个端对端 OCR 的标注，包含了文字的位置信息以及对应的文本信息，`TextRecogCropPacker` 会将文字从图像中裁剪出来，然后再转化成文字识别的统一格式。

标注保存 (Dumper)

`dumper` 来决定要将数据保存为何种格式。目前，MMOCR 支持 `JsonDumper`，`WildreceiptOpensetDumper`，及 `TextRecogLMDBDumper`。他们分别用于将数据保存为标准的 MMOCR Json 格式、Wildreceipt 格式，及文本识别领域学术界常用的 LMDB 格式。

临时文件清理 (Delete)

在处理数据集时，往往会产生一些不需要的临时文件。这里可以以列表的形式传入这些文件或文件夹，在结束转换时即会删除。

生成基础配置 (ConfigGenerator)

为了在数据集准备完毕后可以自动生成基础配置，目前，MMOCR 按任务实现了 TextDetConfigGenerator、TextRecogConfigGenerator 和 TextSpottingConfigGenerator。它们支持的主要参数如下：

在准备好数据集的所有文件后，配置生成器就会自动生成调用该数据集所需要的基础配置文件。下面给出了一个最小化的 TextDetConfigGenerator 配置示例：

```
config_generator = dict(type='TextDetConfigGenerator')
```

生成后的文件默认会被置于 configs/{task}/_base_/datasets/ 下。例如，本例中，icdar 2015 的基础配置文件就会被生成在 configs/textdet/_base_/datasets/icdar2015.py 下：

```
icdar2015_textdet_data_root = 'data/icdar2015'

icdar2015_textdet_train = dict(
    type='OCRDataset',
    data_root=icdar2015_textdet_data_root,
    ann_file='textdet_train.json',
    filter_cfg=dict(filter_empty_gt=True, min_size=32),
    pipeline=None)

icdar2015_textdet_test = dict(
    type='OCRDataset',
    data_root=icdar2015_textdet_data_root,
    ann_file='textdet_test.json',
    test_mode=True,
    pipeline=None)
```

假如数据集比较特殊，标注存在着几个变体，配置生成器也支持在基础配置中生成指向各自变体的变量，但这需要用户在设置时用不同的 dataset_postfix 区分。例如，ICDAR 2015 文字识别数据的测试集就存在着原版和 1811 两种标注版本，可以在 test_anns 中指定它们，如下所示：

```
config_generator = dict(
    type='TextRecogConfigGenerator',
    test_anns=[
        dict(ann_file='textrecog_test.json'),
        dict(dataset_postfix='857', ann_file='textrecog_test_857.json')
    ])
```

配置生成器会生成以下配置：

```
icdar2015_textrecog_data_root = 'data/icdar2015'

```

(下页继续)

(续上页)

```
icdar2015_textrecog_train = dict(
    type='OCRDataset',
    data_root=icdar2015_textrecog_data_root,
    ann_file='textrecog_train.json',
    pipeline=None)

icdar2015_textrecog_test = dict(
    type='OCRDataset',
    data_root=icdar2015_textrecog_data_root,
    ann_file='textrecog_test.json',
    test_mode=True,
    pipeline=None)

icdar2015_1811_textrecog_test = dict(
    type='OCRDataset',
    data_root=icdar2015_textrecog_data_root,
    ann_file='textrecog_test_1811.json',
    test_mode=True,
    pipeline=None)
```

有了该文件后，MMOCR 就能从模型的配置文件中直接导入该数据集到 dataloader 中使用（以下样例节选自 configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py）：

```
_base_ = [
    '../_base_/datasets/icdar2015.py',
    # ...
]

# dataset settings
icdar2015_textdet_train = _base_.icdar2015_textdet_train
icdar2015_textdet_test = _base_.icdar2015_textdet_test
# ...

train_dataloader = dict(
    dataset=icdar2015_textdet_train)

val_dataloader = dict(
    dataset=icdar2015_textdet_test)

test_dataloader = val_dataloader
```

注解： 除非用户在运行脚本的时候手动指定了 overwrite-cfg，配置生成器默认不会自动覆盖已经存在的

基础配置文件。

22.4 向 Dataset Preparer 添加新的数据集

22.4.1 添加公开数据集

MMOCR 已经支持了许多常用的公开数据集。如果你想用的数据集还没有被支持，并且你也愿意为 MMOCR 开源社区贡献代码，你可以按照以下步骤来添加一个新的数据集。

接下来以添加 **ICDAR2013** 数据集为例，展示如何一步一步地添加一个新的公开数据集。

添加 metafile.yml

首先，确认 dataset_zoo/ 中不存在准备添加的数据集。然后我们先新建以待添加数据集命名的文件夹，如 icdar2013/（通常，使用不包含符号的小写英文字母及数字来命名数据集）。在 icdar2013/ 文件夹中，新建 metafile.yml 文件，并按照以下模板来填充数据集的基本信息：

```
Name: 'Incidental Scene Text IC13'
Paper:
  Title: ICDAR 2013 Robust Reading Competition
  URL: https://www.imlab.jp/publication_data/1352/icdar_competition_report.pdf
  Venue: ICDAR
  Year: '2013'
  BibTeX: '@inproceedings{karatzas2013icdar,
  title={ICDAR 2013 robust reading competition},
  author={Karatzas, Dimosthenis and Shafait, Faisal and Uchida, Seiichi and Iwamura,␣
↪Masakazu and i Bigorda, Lluis Gomez and Mestre, Sergi Robles and Mas, Joan and Mota,
↪ David Fernandez and Almazan, Jon Almazan and De Las Heras, Lluis Pere},
  booktitle={2013 12th international conference on document analysis and recognition},
  pages={1484--1493},
  year={2013},
  organization={IEEE}}'
Data:
  Website: https://rrc.cvc.uab.es/?ch=2
  Language:
    - English
  Scene:
    - Natural Scene
  Granularity:
    - Word
  Tasks:
    - textdet
```

(下页继续)

(续上页)

```
  - textrecog
  - textspotting
License:
  Type: N/A
  Link: N/A
Format: .txt
Keywords:
  - Horizontal
```

添加标注示例

最后，可以在 dataset_zoo/icdar2013/ 目录下添加标注示例文件 sample_anno.md 以帮助文档脚本在生成文档时添加标注示例，标注示例文件是一个 Markdown 文件，其内容通常包含了单个样本的原始数据格式。例如，以下代码块展示了 ICDAR2013 数据集的数据样例文件：

````
**Text Detection**

```text
# train split
# x1 y1 x2 y2 "transcript"

158 128 411 181 "Footpath"
443 128 501 169 "To"
64 200 363 243 "Colchester"

# test split
# x1, y1, x2, y2, "transcript"

38, 43, 920, 215, "Tiredness"
275, 264, 665, 450, "kills"
0, 699, 77, 830, "A"
```
````

添加对应任务的配置文件

在 dataset_zoo/icdar2013 中，接着添加以任务名称命名的 .py 配置文件。如 textdet.py，textrecog.py，textspotting.py，kie.py 等。配置模板如下所示：

```
data_root = ''
data_cache = 'data/cache'
train_prepare = dict(
    obtainer=dict(
```

(下页继续)

(续上页)

```
        type='NaiveObtainer',
        data_cache=data_cache,
        files=[
            dict(
                url='xx',
                md5='',
                save_name='xxx',
                mapping=list())
            ]),
    gatherer=dict(type='xxxGatherer', **kwargs),
    parser=dict(type='xxxParser', **kwargs),
    packer=dict(type='TextxxxPacker'), # 对应任务的 Packer
    dumper=dict(type='JsonDumper'),
)
test_prepare = dict(
    obtainer=dict(
        type='NaiveObtainer',
        data_cache=data_cache,
        files=[
            dict(
                url='xx',
                md5='',
                save_name='xxx',
                mapping=list())
            ]),
    gatherer=dict(type='xxxGatherer', **kwargs),
    parser=dict(type='xxxParser', **kwargs),
    packer=dict(type='TextxxxPacker'), # 对应任务的 Packer
    dumper=dict(type='JsonDumper'),
)
```

以文件检测任务为例，来介绍配置文件的具体内容。一般情况下用户无需重新实现新的 obtainer, gatherer, packer 或 dumper, 但是通常需要根据数据集的标注格式实现新的 parser。对于 obtainer 的配置这里不在做过的介绍，可以参考数据集下载、解压、移动。针对 gatherer，通过观察获取的 ICDAR2013 数据集文件发现，其每一张图片都有一个对应的 .txt 格式的标注文件：

```
data_root
├── textdet_imgs/train/
|   ├── img_1.jpg
|   ├── img_2.jpg
|   └── ...
├── annotations/train/
|   ├── gt_img_1.txt
```

(下页继续)

(续上页)

```
|   ├── gt_img_2.txt
|   └── ...
```

且每个标注文件名与图片的对应关系为：gt_img_1.txt 对应 img_1.jpg，以此类推。因此可以使用 PairGatherer 来进行匹配。

```
gatherer=dict(
      type='PairGatherer',
      img_suffixes=['.jpg'],
      rule=[r'(\w+)\.jpg', r'gt_\1.txt'])
```

规则 rule 第一个正则表达式用于匹配图片文件名，第二个正则表达式用于匹配标注文件名。在这里，使用 (\w+) 来匹配图片文件名，使用 gt_\1.txt 来匹配标注文件名，其中 \1 表示第一个正则表达式匹配到的内容。即，实现了将 img_xx.jpg 替换为 gt_img_xx.txt 的功能。

接下来，需要实现 parser，即将原始标注文件解析为标准格式。通常来说，用户在添加新的数据集前，可以浏览已支持数据集的详情页，并查看是否已有相同格式的数据集。如果已有相同格式的数据集，则可以直接使用该数据集的 parser。否则，则需要实现新的格式解析器。

数据格式解析器被统一存储在 mmocr/datasets/preparers/parsers 目录下。所有的 parser 都需要继承 BaseParser，并实现 parse_file 或 parse_files 方法。具体可以参考数据集解析

通过观察 ICDAR2013 数据集的标注文件：

```
158 128 411 181 "Footpath"
443 128 501 169 "To"
64 200 363 243 "Colchester"
542, 710, 938, 841, "break"
87, 884, 457, 1021, "could"
517, 919, 831, 1024, "save"
```

我们发现内置的 ICDARTxtTextDetAnnParser 已经可以满足需求，因此可以直接使用该 parser，并将其配置到 preparer 中。

```
parser=dict(
     type='ICDARTxtTextDetAnnParser',
     remove_strs=[',', '"'],
     encoding='utf-8',
     format='x1 y1 x2 y2 trans',
     separator=' ',
     mode='xyxy')
```

其中，由于标注文件中混杂了多余的引号 “” 和逗号 ,，可以通过指定 remove_strs=[',', '"'] 来进行移除。另外在 format 中指定了标注文件的格式，其中 x1 y1 x2 y2 trans 表示标注文件中的每一行包含了四个坐标和一个文本内容，且坐标和文本内容之间使用空格分隔 (separator=' ')。另外，需要指定 mode

为 xyxy，表示标注文件中的坐标是左上角和右下角的坐标，这样以来，ICDARTxtTextDetAnnParser 即可将该格式的标注解析为统一格式。

对于 packer，以文件检测任务为例，其 packer 为 TextDetPacker，其配置如下：

```
packer=dict(type='TextDetPacker')
```

最后，指定 dumper，这里一般情况下保存为 json 格式，其配置如下：

```
dumper=dict(type='JsonDumper')
```

经过上述配置后，针对 ICDAR2013 训练集的配置文件如下：

```
train_preparer = dict(
    obtainer=dict(
        type='NaiveDataObtainer',
        cache_path=cache_path,
        files=[
            dict(
                url='https://rrc.cvc.uab.es/downloads/'
                'Challenge2_Training_Task12_Images.zip',
                save_name='ic13_textdet_train_img.zip',
                md5='a443b9649fda4229c9bc52751bad08fb',
                content=['image'],
                mapping=[['ic13_textdet_train_img', 'textdet_imgs/train']]),
            dict(
                url='https://rrc.cvc.uab.es/downloads/'
                'Challenge2_Training_Task1_GT.zip',
                save_name='ic13_textdet_train_gt.zip',
                md5='f3a425284a66cd67f455d389c972cce4',
                content=['annotation'],
                mapping=[['ic13_textdet_train_gt', 'annotations/train']]),
        ]),
    gatherer=dict(
        type='PairGatherer',
        img_suffixes=['.jpg'],
        rule=[r'(\w+)\.jpg', r'gt_\1.txt']),
    parser=dict(
        type='ICDARTxtTextDetAnnParser',
        remove_strs=[',', '"'],
        format='x1 y1 x2 y2 trans',
        separator=' ',
        mode='xyxy'),
    packer=dict(type='TextDetPacker'),
    dumper=dict(type='JsonDumper'),
)
```

为了在数据集准备完毕后可以自动生成基础配置，还需要配置一下对应任务的 config_generator。

在本例中，因为为文字检测任务，仅需要设置 Generator 为 TextDetConfigGenerator 即可

```
config_generator = dict(type='TextDetConfigGenerator', )
```

22.4.2 添加私有数据集

待更新…

CHAPTER 23

Text Detection

注解： This page is a manual preparation guide for datasets not yet supported by *Dataset Preparer*, which all these scripts will be eventually migrated into.

23.1 Overview

23.1.1 Install AWS CLI (optional)

- Since there are some datasets that require the AWS CLI to be installed in advance, we provide a quick installation guide here:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip
↪"
unzip awscliv2.zip
sudo ./aws/install
./aws/install -i /usr/local/aws-cli -b /usr/local/bin
!aws configure
# this command will require you to input keys, you can skip them except
# for the Default region name
# AWS Access Key ID [None]:
# AWS Secret Access Key [None]:
```

(下页继续)

(续上页)

```
# Default region name [None]: us-east-1
# Default output format [None]
```

For users in China, these datasets can also be downloaded from OpenDataLab with high speed:

- CTW1500
- ICDAR2013
- ICDAR2015
- Totaltext
- MSRA-TD500

23.2 Important Note

注解： **For users who want to train models on CTW1500, ICDAR 2015/2017, and Totaltext dataset,** there might be some images containing orientation info in EXIF data. The default OpenCV backend used in MMCV would read them and apply the rotation on the images. However, their gold annotations are made on the raw pixels, and such inconsistency results in false examples in the training set. Therefore, users should use `dict(type='LoadImageFromFile', color_type='color_ignore_orientation')` in pipelines to change MMCV's default loading behaviour. (see DBNet's pipeline config for example)

23.3 ICDAR 2011 (Born-Digital Images)

- Step1: Download `Challenge1_Training_Task12_Images.zip`, `Challenge1_Training_Task1_GT.zip`, `Challenge1_Test_Task12_Images.zip`, and `Challenge1_Test_Task1_GT.zip` from homepage `Task 1.1: Text Localization (2013 edition)`.

```
mkdir icdar2011 && cd icdar2011
mkdir imgs && mkdir annotations

# Download ICDAR 2011
wget https://rrc.cvc.uab.es/downloads/Challenge1_Training_Task12_Images.zip --no-
↪check-certificate
wget https://rrc.cvc.uab.es/downloads/Challenge1_Training_Task1_GT.zip --no-check-
↪certificate
wget https://rrc.cvc.uab.es/downloads/Challenge1_Test_Task12_Images.zip --no-
↪check-certificate
wget https://rrc.cvc.uab.es/downloads/Challenge1_Test_Task1_GT.zip --no-check-
↪certificate
```

(下页继续)

(续上页)

```
# For images
unzip -q Challenge1_Training_Task12_Images.zip -d imgs/training
unzip -q Challenge1_Test_Task12_Images.zip -d imgs/test
# For annotations
unzip -q Challenge1_Training_Task1_GT.zip -d annotations/training
unzip -q Challenge1_Test_Task1_GT.zip -d annotations/test

rm Challenge1_Training_Task12_Images.zip && rm Challenge1_Test_Task12_Images.zip &
↪& rm Challenge1_Training_Task1_GT.zip && rm Challenge1_Test_Task1_GT.zip
```

- Step 2: Generate `instances_training.json` and `instances_test.json` with the following command:

```
python tools/dataset_converters/textdet/ic11_converter.py PATH/TO/icdar2011 --
↪nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── icdar2011
|   ├── imgs
|   ├── instances_test.json
|   └── instances_training.json
```

23.4 ICDAR 2017

- Follow similar steps as ICDAR 2015.
- The resulting directory structure looks like the following:

```
├── icdar2017
|   ├── imgs
|   ├── annotations
|   ├── instances_training.json
|   └── instances_val.json
```

23.5 CurvedSynText150k

- Step1: Download syntext1.zip and syntext2.zip to `CurvedSynText150k/`.
- Step2:

```
unzip -q syntext1.zip
mv train.json train1.json
unzip images.zip
rm images.zip

unzip -q syntext2.zip
mv train.json train2.json
unzip images.zip
rm images.zip
```

- Step3: Download instances_training.json to `CurvedSynText150k/`
- Or, generate `instances_training.json` with following command:

```
python tools/dataset_converters/common/curvedsyntext_converter.py PATH/TO/
↪CurvedSynText150k --nproc 4
```

- The resulting directory structure looks like the following:

```
├── CurvedSynText150k
|   ├── syntext_word_eng
|   ├── emcs_imgs
|   └── instances_training.json
```

23.6 DeText

- Step1: Download `ch9_training_images.zip`, `ch9_training_localization_transcription_gt.zip`, `ch9_validation_images.zip`, and `ch9_validation_localization_transcription_gt.zip` from **Task 3: End to End** on the homepage.

```
mkdir detext && cd detext
mkdir imgs && mkdir annotations && mkdir imgs/training && mkdir imgs/val && mkdir␣
↪annotations/training && mkdir annotations/val

# Download DeText
wget https://rrc.cvc.uab.es/downloads/ch9_training_images.zip --no-check-
↪certificate
wget https://rrc.cvc.uab.es/downloads/ch9_training_localization_transcription_gt.
↪zip --no-check-certificate
```

(下页继续)

(续上页)

```
wget https://rrc.cvc.uab.es/downloads/ch9_validation_images.zip --no-check-
↪certificate
wget https://rrc.cvc.uab.es/downloads/ch9_validation_localization_transcription_
↪gt.zip --no-check-certificate

# Extract images and annotations
unzip -q ch9_training_images.zip -d imgs/training && unzip -q ch9_training_
↪localization_transcription_gt.zip -d annotations/training && unzip -q ch9_
↪validation_images.zip -d imgs/val && unzip -q ch9_validation_localization_
↪transcription_gt.zip -d annotations/val

# Remove zips
rm ch9_training_images.zip && rm ch9_training_localization_transcription_gt.zip &&
↪ rm ch9_validation_images.zip && rm ch9_validation_localization_transcription_
↪gt.zip
```

- Step2: Generate `instances_training.json` and `instances_val.json` with following command:

```
python tools/dataset_converters/textdet/detext_converter.py PATH/TO/detext --
↪nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── detext
|   ├── annotations
|   ├── imgs
|   ├── instances_test.json
|   └── instances_training.json
```

23.7 Lecture Video DB

- Step1: Download IIIT-CVid.zip to `lv/`.

```
mkdir lv && cd lv

# Download LV dataset
wget http://cdn.iiit.ac.in/cdn/preon.iiit.ac.in/~kartik/IIIT-CVid.zip
unzip -q IIIT-CVid.zip

mv IIIT-CVid/Frames imgs

rm IIIT-CVid.zip
```

- Step2: Generate `instances_training.json`, `instances_val.json`, and `instances_test.json` with following command:

```
python tools/dataset_converters/textdet/lv_converter.py PATH/TO/lv --nproc 4
```

- The resulting directory structure looks like the following:

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

23.8 LSVT

- Step1: Download train_full_images_0.tar.gz, train_full_images_1.tar.gz, and train_full_labels.json to `lsvt/`.

```
mkdir lsvt && cd lsvt

# Download LSVT dataset
wget https://dataset-bj.cdn.bcebos.com/lsvt/train_full_images_0.tar.gz
wget https://dataset-bj.cdn.bcebos.com/lsvt/train_full_images_1.tar.gz
wget https://dataset-bj.cdn.bcebos.com/lsvt/train_full_labels.json

mkdir annotations
tar -xf train_full_images_0.tar.gz && tar -xf train_full_images_1.tar.gz
mv train_full_labels.json annotations/ && mv train_full_images_1/*.jpg train_full_
↪images_0/
mv train_full_images_0 imgs

rm train_full_images_0.tar.gz && rm train_full_images_1.tar.gz && rm -rf train_
↪full_images_1
```

- Step2: Generate `instances_training.json` and `instances_val.json` (optional) with the following command:

```
# Annotations of LSVT test split is not publicly available, split a validation
# set by adding --val-ratio 0.2
python tools/dataset_converters/textdet/lsvt_converter.py PATH/TO/lsvt
```

- After running the above codes, the directory structure should be as follows:

```
|── lsvt
|    ├── imgs
|    ├── instances_training.json
|    └── instances_val.json (optional)
```

23.9 IMGUR

- Step1: Run `download_imgur5k.py` to download images. You can merge PR#5 in your local repository to enable a **much faster** parallel execution of image download.

```
mkdir imgur && cd imgur

git clone https://github.com/facebookresearch/IMGUR5K-Handwriting-Dataset.git

# Download images from imgur.com. This may take SEVERAL HOURS!
python ./IMGUR5K-Handwriting-Dataset/download_imgur5k.py --dataset_info_dir ./
↪IMGUR5K-Handwriting-Dataset/dataset_info/ --output_dir ./imgs

# For annotations
mkdir annotations
mv ./IMGUR5K-Handwriting-Dataset/dataset_info/*.json annotations

rm -rf IMGUR5K-Handwriting-Dataset
```

- Step2: Generate `instances_train.json`, `instance_val.json` and `instances_test.json` with the following command:

```
python tools/dataset_converters/textdet/imgur_converter.py PATH/TO/imgur
```

- After running the above codes, the directory structure should be as follows:

```
|── imgur
|    ├── annotations
|    ├── imgs
|    ├── instances_test.json
|    ├── instances_training.json
|    └── instances_val.json
```

23.10 KAIST

- Step1: Complete download KAIST_all.zip to `kaist/`.

```
mkdir kaist && cd kaist
mkdir imgs && mkdir annotations

# Download KAIST dataset
wget http://www.iapr-tc11.org/dataset/KAIST_SceneText/KAIST_all.zip
unzip -q KAIST_all.zip

rm KAIST_all.zip
```

- Step2: Extract zips:

```
python tools/dataset_converters/common/extract_kaist.py PATH/TO/kaist
```

- Step3: Generate `instances_training.json` and `instances_val.json` (optional) with following command:

```
# Since KAIST does not provide an official split, you can split the dataset by␣
↪adding --val-ratio 0.2
python tools/dataset_converters/textdet/kaist_converter.py PATH/TO/kaist --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── kaist
|   ├── annotations
|   ├── imgs
|   ├── instances_training.json
|   └── instances_val.json (optional)
```

23.11 MTWI

- Step1: Download `mtwi_2018_train.zip` from homepage.

```
mkdir mtwi && cd mtwi

unzip -q mtwi_2018_train.zip
mv image_train imgs && mv txt_train annotations

rm mtwi_2018_train.zip
```

- Step2: Generate `instances_training.json` and `instance_val.json` (optional) with the following command:

```
# Annotations of MTWI test split is not publicly available, split a validation
# set by adding --val-ratio 0.2
python tools/dataset_converters/textdet/mtwi_converter.py PATH/TO/mtwi --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── mtwi
|   ├── annotations
|   ├── imgs
|   ├── instances_training.json
|   └── instances_val.json (optional)
```

23.12 ReCTS

- Step1: Download ReCTS.zip to `rects/` from the homepage.

```
mkdir rects && cd rects

# Download ReCTS dataset
# You can also find Google Drive link on the dataset homepage
wget https://datasets.cvc.uab.es/rrc/ReCTS.zip --no-check-certificate
unzip -q ReCTS.zip

mv img imgs && mv gt_unicode annotations

rm ReCTS.zip && rm -rf gt
```

- Step2: Generate `instances_training.json` and `instances_val.json` (optional) with following command:

```
# Annotations of ReCTS test split is not publicly available, split a validation
# set by adding --val-ratio 0.2
python tools/dataset_converters/textdet/rects_converter.py PATH/TO/rects --nproc␣
↪4 --val-ratio 0.2
```

- After running the above codes, the directory structure should be as follows:

```
|── rects
|   ├── annotations
|   ├── imgs
```

(下页继续)

(续上页)

```
|   ├── instances_val.json (optional)
|   └── instances_training.json
```

23.13 ILST

- Step1: Download `IIIT-ILST` from onedrive
- Step2: Run the following commands

```
unzip -q IIIT-ILST.zip && rm IIIT-ILST.zip
cd IIIT-ILST

# rename files
cd Devanagari && for i in `ls`; do mv -f $i `echo "devanagari_"$i`; done && cd ..
cd Malayalam && for i in `ls`; do mv -f $i `echo "malayalam_"$i`; done && cd ..
cd Telugu && for i in `ls`; do mv -f $i `echo "telugu_"$i`; done && cd ..

# transfer image path
mkdir imgs && mkdir annotations
mv Malayalam/{*jpg,*jpeg} imgs/ && mv Malayalam/*xml annotations/
mv Devanagari/*jpg imgs/ && mv Devanagari/*xml annotations/
mv Telugu/*jpeg imgs/ && mv Telugu/*xml annotations/

# remove unnecessary files
rm -rf Devanagari && rm -rf Malayalam && rm -rf Telugu && rm -rf README.txt
```

- Step3: Generate `instances_training.json` and `instances_val.json` (optional). Since the original dataset doesn't have a validation set, you may specify `--val-ratio` to split the dataset. E.g., if val-ratio is 0.2, then 20% of the data are left out as the validation set in this example.

```
python tools/dataset_converters/textdet/ilst_converter.py    PATH/TO/IIIT-ILST --
↪nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── IIIT-ILST
|   ├── annotations
|   ├── imgs
|   ├── instances_val.json (optional)
|   └── instances_training.json
```

23.14 VinText

- Step1: Download vintext.zip to `vintext`

```
mkdir vintext && cd vintext

# Download dataset from google drive
wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&
↪confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --
↪no-check-certificate 'https://docs.google.com/uc?export=download&
↪id=1UUQhNvzgpZy7zXBFQp0Qox-BBjunZ0ml' -O- | sed -rn 's/.*confirm=([0-9A-Za-z_
↪]+).*/\1\n/p')&id=1UUQhNvzgpZy7zXBFQp0Qox-BBjunZ0ml" -O vintext.zip && rm -rf /
↪tmp/cookies.txt

# Extract images and annotations
unzip -q vintext.zip && rm vintext.zip
mv vietnamese/labels ./ && mv vietnamese/test_image ./ && mv vietnamese/train_
↪images ./ && mv vietnamese/unseen_test_images ./
rm -rf vietnamese

# Rename files
mv labels annotations && mv test_image test && mv train_images  training && mv␣
↪unseen_test_images  unseen_test
mkdir imgs
mv training imgs/ && mv test imgs/ && mv unseen_test imgs/
```

- Step2: Generate `instances_training.json`, `instances_test.json` and `instances_unseen_test.json`

```
python tools/dataset_converters/textdet/vintext_converter.py PATH/TO/vintext --
↪nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── vintext
|   ├── annotations
|   ├── imgs
|   ├── instances_test.json
|   ├── instances_unseen_test.json
|   └── instances_training.json
```

23.15 BID

- Step1: Download BID Dataset.zip
- Step2: Run the following commands to preprocess the dataset

```
# Rename
mv BID\ Dataset.zip BID_Dataset.zip

# Unzip and Rename
unzip -q BID_Dataset.zip && rm BID_Dataset.zip
mv BID\ Dataset BID

# The BID dataset has a problem of permission, and you may
# add permission for this file
chmod -R 777 BID
cd BID
mkdir imgs && mkdir annotations

# For images and annotations
mv CNH_Aberta/*in.jpg imgs && mv CNH_Aberta/*txt annotations && rm -rf CNH_Aberta
mv CNH_Frente/*in.jpg imgs && mv CNH_Frente/*txt annotations && rm -rf CNH_Frente
mv CNH_Verso/*in.jpg imgs && mv CNH_Verso/*txt annotations && rm -rf CNH_Verso
mv CPF_Frente/*in.jpg imgs && mv CPF_Frente/*txt annotations && rm -rf CPF_Frente
mv CPF_Verso/*in.jpg imgs && mv CPF_Verso/*txt annotations && rm -rf CPF_Verso
mv RG_Aberto/*in.jpg imgs && mv RG_Aberto/*txt annotations && rm -rf RG_Aberto
mv RG_Frente/*in.jpg imgs && mv RG_Frente/*txt annotations && rm -rf RG_Frente
mv RG_Verso/*in.jpg imgs && mv RG_Verso/*txt annotations && rm -rf RG_Verso

# Remove unnecessary files
rm -rf desktop.ini
```

- Step3: - Step3: Generate `instances_training.json` and `instances_val.json` (optional). Since the original dataset doesn't have a validation set, you may specify `--val-ratio` to split the dataset. E.g., if val-ratio is 0.2, then 20% of the data are left out as the validation set in this example.

```
python tools/dataset_converters/textdet/bid_converter.py PATH/TO/BID --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── BID
|   ├── annotations
|   ├── imgs
|   ├── instances_training.json
|   └── instances_val.json (optional)
```

23.16 RCTW

- Step1: Download `train_images.zip.001`, `train_images.zip.002`, and `train_gts.zip` from the homepage, extract the zips to `rctw/imgs` and `rctw/annotations`, respectively.
- Step2: Generate `instances_training.json` and `instances_val.json` (optional). Since the test annotations are not publicly available, you may specify `--val-ratio` to split the dataset. E.g., if val-ratio is 0.2, then 20% of the data are left out as the validation set in this example.

```
# Annotations of RCTW test split is not publicly available, split a validation␣
↪set by adding --val-ratio 0.2
python tools/dataset_converters/textdet/rctw_converter.py PATH/TO/rctw --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── rctw
|   ├── annotations
|   ├── imgs
|   ├── instances_training.json
|   └── instances_val.json (optional)
```

23.17 HierText

- Step1 (optional): Install AWS CLI.
- Step2: Clone HierText repo to get annotations

```
mkdir HierText
git clone https://github.com/google-research-datasets/hiertext.git
```

- Step3: Download `train.tgz`, `validation.tgz` from aws

```
aws s3 --no-sign-request cp s3://open-images-dataset/ocr/train.tgz .
aws s3 --no-sign-request cp s3://open-images-dataset/ocr/validation.tgz .
```

- Step4: Process raw data

```
# process annotations
mv hiertext/gt ./
rm -rf hiertext
mv gt annotations
gzip -d annotations/train.jsonl.gz
gzip -d annotations/validation.jsonl.gz
# process images
```

(下页继续)

(续上页)

```
mkdir imgs
mv train.tgz imgs/
mv validation.tgz imgs/
tar -xzvf imgs/train.tgz
tar -xzvf imgs/validation.tgz
```

- Step5: Generate `instances_training.json` and `instance_val.json`. HierText includes different levels of annotation, from paragraph, line, to word. Check the original paper for details. E.g. set `--level paragraph` to get paragraph-level annotation. Set `--level line` to get line-level annotation. set `--level word` to get word-level annotation.

```
# Collect word annotation from HierText  --level word
python tools/dataset_converters/textdet/hiertext_converter.py PATH/TO/HierText --
↪level word --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── HierText
|   ├── annotations
|   ├── imgs
|   ├── instances_training.json
|   └── instances_val.json
```

23.18 ArT

- Step1: Download `train_images.tar.gz`, and `train_labels.json` from the homepage to `art/`

```
mkdir art && cd art
mkdir annotations

# Download ArT dataset
wget https://dataset-bj.cdn.bcebos.com/art/train_images.tar.gz --no-check-
↪certificate
wget https://dataset-bj.cdn.bcebos.com/art/train_labels.json --no-check-
↪certificate

# Extract
tar -xf train_images.tar.gz
mv train_images imgs
mv train_labels.json annotations/
```

(下页继续)

(续上页)

```
# Remove unnecessary files
rm train_images.tar.gz
```

- Step2: Generate `instances_training.json` and `instances_val.json` (optional). Since the test annotations are not publicly available, you may specify `--val-ratio` to split the dataset. E.g., if val-ratio is 0.2, then 20% of the data are left out as the validation set in this example.

```
# Annotations of ArT test split is not publicly available, split a validation set␣
↪by adding --val-ratio 0.2
python tools/data/textdet/art_converter.py PATH/TO/art --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
|── art
|   ├── annotations
|   ├── imgs
|   ├── instances_training.json
|   └── instances_val.json (optional)
```

CHAPTER 24

Text Recognition

注解： This page is a manual preparation guide for datasets not yet supported by *Dataset Preparer*, which all these scripts will be eventually migrated into.

24.1 Overview

(*) Since the official homepage is unavailable now, we provide an alternative for quick reference. However, we do not guarantee the correctness of the dataset.

24.1.1 Install AWS CLI (optional)

- Since there are some datasets that require the AWS CLI to be installed in advance, we provide a quick installation guide here:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip
↪"
unzip awscliv2.zip
sudo ./aws/install
./aws/install -i /usr/local/aws-cli -b /usr/local/bin
!aws configure
# this command will require you to input keys, you can skip them except
```

(下页继续)

(续上页)

```
# for the Default region name
# AWS Access Key ID [None]:
# AWS Secret Access Key [None]:
# Default region name [None]: us-east-1
# Default output format [None]
```

For users in China, these datasets can also be downloaded from OpenDataLab with high speed:

- icdar_2013
- icdar_2015
- IIIT5K
- ct80
- svt
- Totaltext
- IAM

24.2 ICDAR 2011 (Born-Digital Images)

- Step1: Download `Challenge1_Training_Task3_Images_GT.zip`, `Challenge1_Test_Task3_Images.zip`, and `Challenge1_Test_Task3_GT.txt` from homepage `Task 1.3: Word Recognition (2013 edition)`.

```
mkdir icdar2011 && cd icdar2011
mkdir annotations

# Download ICDAR 2011
wget https://rrc.cvc.uab.es/downloads/Challenge1_Training_Task3_Images_GT.zip --
↪no-check-certificate
wget https://rrc.cvc.uab.es/downloads/Challenge1_Test_Task3_Images.zip --no-check-
↪certificate
wget https://rrc.cvc.uab.es/downloads/Challenge1_Test_Task3_GT.txt --no-check-
↪certificate

# For images
mkdir crops
unzip -q Challenge1_Training_Task3_Images_GT.zip -d crops/train
unzip -q Challenge1_Test_Task3_Images.zip -d crops/test
```

(下页继续)

(续上页)

```
# For annotations
mv Challenge1_Test_Task3_GT.txt annotations && mv crops/train/gt.txt annotations/
↪Challenge1_Train_Task3_GT.txt
```

- Step2: Convert original annotations to `train_labels.json` and `test_labels.json` with the following command:

```
python tools/dataset_converters/textrecog/ic11_converter.py PATH/TO/icdar2011
```

- After running the above codes, the directory structure should be as follows:

```
├── icdar2011
|   ├── crops
|   ├── train_labels.json
|   └── test_labels.json
```

24.3 coco_text

- Step1: Download from homepage
- Step2: Download train_labels.json
- After running the above codes, the directory structure should be as follows:

```
├── coco_text
|   ├── train_labels.json
|   └── train_words
```

24.4 SynthAdd

- Step1: Download `SynthText_Add.zip` from SynthAdd (code:627x))
- Step2: Download train_labels.json
- Step3:

```
mkdir SynthAdd && cd SynthAdd

mv /path/to/SynthText_Add.zip .

unzip SynthText_Add.zip
```

(下页继续)

(续上页)

```
mv /path/to/train_labels.json .

# create soft link
cd /path/to/mmocr/data/recog

ln -s /path/to/SynthAdd SynthAdd
```

- After running the above codes, the directory structure should be as follows:

```
├── SynthAdd
|   ├── train_labels.json
|   └── SynthText_Add
```

24.5 OpenVINO

- Step1 (optional): Install AWS CLI.
- Step2: Download Open Images subsets `train_1`, `train_2`, `train_5`, `train_f`, and `validation` to `openvino/`.

```
mkdir openvino && cd openvino

# Download Open Images subsets
for s in 1 2 5 f; do
  aws s3 --no-sign-request cp s3://open-images-dataset/tar/train_${s}.tar.gz .
done
aws s3 --no-sign-request cp s3://open-images-dataset/tar/validation.tar.gz .

# Download annotations
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

# Extract images
mkdir -p openimages_v5/val
for s in 1 2 5 f; do
  tar zxf train_${s}.tar.gz -C openimages_v5
```

(下页继续)

(续上页)

```
done
tar zxf validation.tar.gz -C openimages_v5/val
```

- Step3: Generate `train_{1,2,5,f}_labels.json`, `val_labels.json` and crop images using 4 processes with the following command:

```
python tools/dataset_converters/textrecog/openvino_converter.py /path/to/openvino␣
↪4
```

- After running the above codes, the directory structure should be as follows:

```
├── OpenVINO
|   ├── image_1
|   ├── image_2
|   ├── image_5
|   ├── image_f
|   ├── image_val
|   ├── train_1_labels.json
|   ├── train_2_labels.json
|   ├── train_5_labels.json
|   ├── train_f_labels.json
|   └── val_labels.json
```

24.6 DeText

- Step1: Download `ch9_training_images.zip`, `ch9_training_localization_transcription_gt.zip`, `ch9_validation_images.zip`, and `ch9_validation_localization_transcription_gt.zip` from **Task 3: End to End** on the homepage.

```
mkdir detext && cd detext
mkdir imgs && mkdir annotations && mkdir imgs/training && mkdir imgs/val && mkdir␣
↪annotations/training && mkdir annotations/val

# Download DeText
wget https://rrc.cvc.uab.es/downloads/ch9_training_images.zip --no-check-
↪certificate
wget https://rrc.cvc.uab.es/downloads/ch9_training_localization_transcription_gt.
↪zip --no-check-certificate
wget https://rrc.cvc.uab.es/downloads/ch9_validation_images.zip --no-check-
↪certificate
wget https://rrc.cvc.uab.es/downloads/ch9_validation_localization_transcription_
↪gt.zip --no-check-certificate
```

(下页继续)

(续上页)

```
# Extract images and annotations
unzip -q ch9_training_images.zip -d imgs/training && unzip -q ch9_training_
↪localization_transcription_gt.zip -d annotations/training && unzip -q ch9_
↪validation_images.zip -d imgs/val && unzip -q ch9_validation_localization_
↪transcription_gt.zip -d annotations/val

# Remove zips
rm ch9_training_images.zip && rm ch9_training_localization_transcription_gt.zip &&
↪ rm ch9_validation_images.zip && rm ch9_validation_localization_transcription_
↪gt.zip
```

- Step2: Generate `train_labels.json` and `test_labels.json` with following command:

```
# Add --preserve-vertical to preserve vertical texts for training, otherwise
# vertical images will be filtered and stored in PATH/TO/detext/ignores
python tools/dataset_converters/textrecog/detext_converter.py PATH/TO/detext --
↪nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── detext
|   ├── crops
|   ├── ignores
|   ├── train_labels.json
|   └── test_labels.json
```

24.7 NAF

- Step1: Download labeled_images.tar.gz to `naf/`.

```
mkdir naf && cd naf

# Download NAF dataset
wget https://github.com/herobd/NAF_dataset/releases/download/v1.0/labeled_images.
↪tar.gz
tar -zxf labeled_images.tar.gz

# For images
mkdir annotations && mv labeled_images imgs

# For annotations
```

(下页继续)

(续上页)

```
git clone https://github.com/herobd/NAF_dataset.git
mv NAF_dataset/train_valid_test_split.json annotations/ && mv NAF_dataset/groups␣
↪annotations/

rm -rf NAF_dataset && rm labeled_images.tar.gz
```

- Step2: Generate `train_labels.json`, `val_labels.json`, and `test_labels.json` with following command:

```
# Add --preserve-vertical to preserve vertical texts for training, otherwise
# vertical images will be filtered and stored in PATH/TO/naf/ignores
python tools/dataset_converters/textrecog/naf_converter.py PATH/TO/naf --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── naf
│   ├── crops
│   ├── train_labels.json
│   ├── val_labels.json
│   └── test_labels.json
```

24.8 Lecture Video DB

警告： This section is not fully tested yet.

注解： The LV dataset has already provided cropped images and the corresponding annotations

- Step1: Download IIIT-CVid.zip to `lv/`.

```
mkdir lv && cd lv

# Download LV dataset
wget http://cdn.iiit.ac.in/cdn/preon.iiit.ac.in/~kartik/IIIT-CVid.zip
unzip -q IIIT-CVid.zip

# For image
mv IIIT-CVid/Crops ./
```

(下页继续)

(续上页)

```
# For annotation
mv IIIT-CVid/train.txt train_labels.json && mv IIIT-CVid/val.txt val_label.txt &&␣
↪mv IIIT-CVid/test.txt test_labels.json

rm IIIT-CVid.zip
```

- Step2: Generate `train_labels.json`, `val.json`, and `test.json` with following command:

```
python tools/dataset_converters/textdreog/lv_converter.py PATH/TO/lv
```

- After running the above codes, the directory structure should be as follows:

```
├── lv
|   ├── Crops
|   ├── train_labels.json
|   └── test_labels.json
```

24.9 LSVT

警告: This section is not fully tested yet.

- Step1: Download train_full_images_0.tar.gz, train_full_images_1.tar.gz, and train_full_labels.json to `lsvt/`.

```
mkdir lsvt && cd lsvt

# Download LSVT dataset
wget https://dataset-bj.cdn.bcebos.com/lsvt/train_full_images_0.tar.gz
wget https://dataset-bj.cdn.bcebos.com/lsvt/train_full_images_1.tar.gz
wget https://dataset-bj.cdn.bcebos.com/lsvt/train_full_labels.json

mkdir annotations
tar -xf train_full_images_0.tar.gz && tar -xf train_full_images_1.tar.gz
mv train_full_labels.json annotations/ && mv train_full_images_1/*.jpg train_full_
↪images_0/
mv train_full_images_0 imgs

rm train_full_images_0.tar.gz && rm train_full_images_1.tar.gz && rm -rf train_
↪full_images_1
```

- Step2: Generate `train_labels.json` and `val_label.json` (optional) with the following command:

```
# Annotations of LSVT test split is not publicly available, split a validation
# set by adding --val-ratio 0.2
# Add --preserve-vertical to preserve vertical texts for training, otherwise
# vertical images will be filtered and stored in PATH/TO/lsvt/ignores
python tools/dataset_converters/textdrecog/lsvt_converter.py PATH/TO/lsvt --nproc␣
↪4
```

- After running the above codes, the directory structure should be as follows:

```
├── lsvt
|   ├── crops
|   ├── ignores
|   ├── train_labels.json
|   └── val_label.json (optional)
```

24.10 IMGUR

警告: This section is not fully tested yet.

- Step1: Run `download_imgur5k.py` to download images. You can merge PR#5 in your local repository to enable a **much faster** parallel execution of image download.

```
mkdir imgur && cd imgur

git clone https://github.com/facebookresearch/IMGUR5K-Handwriting-Dataset.git

# Download images from imgur.com. This may take SEVERAL HOURS!
python ./IMGUR5K-Handwriting-Dataset/download_imgur5k.py --dataset_info_dir ./
↪IMGUR5K-Handwriting-Dataset/dataset_info/ --output_dir ./imgs

# For annotations
mkdir annotations
mv ./IMGUR5K-Handwriting-Dataset/dataset_info/*.json annotations

rm -rf IMGUR5K-Handwriting-Dataset
```

- Step2: Generate `train_labels.json`, `val_label.txt` and `test_labels.json` and crop images with the following command:

```
python tools/dataset_converters/textrecog/imgur_converter.py PATH/TO/imgur
```

- After running the above codes, the directory structure should be as follows:

```
├── imgur
│   ├── crops
│   ├── train_labels.json
│   ├── test_labels.json
│   └── val_label.json
```

24.11 KAIST

警告: This section is not fully tested yet.

- Step1: Download KAIST_all.zip to `kaist/`.

```
mkdir kaist && cd kaist
mkdir imgs && mkdir annotations

# Download KAIST dataset
wget http://www.iapr-tc11.org/dataset/KAIST_SceneText/KAIST_all.zip
unzip -q KAIST_all.zip && rm KAIST_all.zip
```

- Step2: Extract zips:

```
python tools/dataset_converters/common/extract_kaist.py PATH/TO/kaist
```

- Step3: Generate `train_labels.json` and `val_label.json` (optional) with following command:

```
# Since KAIST does not provide an official split, you can split the dataset by␣
↪adding --val-ratio 0.2
# Add --preserve-vertical to preserve vertical texts for training, otherwise
# vertical images will be filtered and stored in PATH/TO/kaist/ignores
python tools/dataset_converters/textrecog/kaist_converter.py PATH/TO/kaist --
↪nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── kaist
│   ├── crops
│   ├── ignores
│   ├── train_labels.json
│   └── val_label.json (optional)
```

24.12 MTWI

警告： This section is not fully tested yet.

- Step1: Download `mtwi_2018_train.zip` from homepage.

```
mkdir mtwi && cd mtwi

unzip -q mtwi_2018_train.zip
mv image_train imgs && mv txt_train annotations

rm mtwi_2018_train.zip
```

- Step2: Generate `train_labels.json` and `val_label.json` (optional) with the following command:

```
# Annotations of MTWI test split is not publicly available, split a validation
# set by adding --val-ratio 0.2
# Add --preserve-vertical to preserve vertical texts for training, otherwise
# vertical images will be filtered and stored in PATH/TO/mtwi/ignores
python tools/dataset_converters/textrecog/mtwi_converter.py PATH/TO/mtwi --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── mtwi
|   ├── crops
|   ├── train_labels.json
|   └── val_label.json (optional)
```

24.13 ReCTS

警告： This section is not fully tested yet.

- Step1: Download ReCTS.zip to `rects/` from the homepage.

```
mkdir rects && cd rects

# Download ReCTS dataset
# You can also find Google Drive link on the dataset homepage
wget https://datasets.cvc.uab.es/rrc/ReCTS.zip --no-check-certificate
```

(下页继续)

(续上页)

```
unzip -q ReCTS.zip

mv img imgs && mv gt_unicode annotations

rm ReCTS.zip -f && rm -rf gt
```

- Step2: Generate `train_labels.json` and `val_label.json` (optional) with the following command:

```
# Annotations of ReCTS test split is not publicly available, split a validation
# set by adding --val-ratio 0.2
# Add --preserve-vertical to preserve vertical texts for training, otherwise
# vertical images will be filtered and stored in PATH/TO/rects/ignores
python tools/dataset_converters/textrecog/rects_converter.py PATH/TO/rects --
↪nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── rects
|   ├── crops
|   ├── ignores
|   ├── train_labels.json
|   └── val_label.json (optional)
```

24.14 ILST

警告: This section is not fully tested yet.

- Step1: Download `IIIT-ILST.zip` from onedrive link
- Step2: Run the following commands

```
unzip -q IIIT-ILST.zip && rm IIIT-ILST.zip
cd IIIT-ILST

# rename files
cd Devanagari && for i in `ls`; do mv -f $i `echo "devanagari_"$i`; done && cd ..
cd Malayalam && for i in `ls`; do mv -f $i `echo "malayalam_"$i`; done && cd ..
cd Telugu && for i in `ls`; do mv -f $i `echo "telugu_"$i`; done && cd ..

# transfer image path
mkdir imgs && mkdir annotations
```

(下页继续)

(续上页)

```
mv Malayalam/{*jpg,*jpeg} imgs/ && mv Malayalam/*xml annotations/
mv Devanagari/*jpg imgs/ && mv Devanagari/*xml annotations/
mv Telugu/*jpeg imgs/ && mv Telugu/*xml annotations/

# remove unnecessary files
rm -rf Devanagari && rm -rf Malayalam && rm -rf Telugu && rm -rf README.txt
```

- Step3: Generate `train_labels.json` and `val_label.json` (optional) and crop images using 4 processes with the following command (add `--preserve-vertical` if you wish to preserve the images containing vertical texts). Since the original dataset doesn't have a validation set, you may specify `--val-ratio` to split the dataset. E.g., if val-ratio is 0.2, then 20% of the data are left out as the validation set in this example.

```
python tools/dataset_converters/textrecog/ilst_converter.py PATH/TO/IIIT-ILST --
↪nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── IIIT-ILST
|   ├── crops
|   ├── ignores
|   ├── train_labels.json
|   └── val_label.json (optional)
```

24.15 VinText

警告: This section is not fully tested yet.

- Step1: Download vintext.zip to `vintext`

```
mkdir vintext && cd vintext

# Download dataset from google drive
wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&
↪confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --
↪no-check-certificate 'https://docs.google.com/uc?export=download&
↪id=1UUQhNvzgpZy7zXBFQp0Qox-BBjunZ0ml' -O- | sed -rn 's/.*confirm=([0-9A-Za-z_
↪]+).*/\1\n/p')&id=1UUQhNvzgpZy7zXBFQp0Qox-BBjunZ0ml" -O vintext.zip && rm -rf /
↪tmp/cookies.txt

# Extract images and annotations
```

(下页继续)

(续上页)

```
unzip -q vintext.zip && rm vintext.zip
mv vietnamese/labels ./ && mv vietnamese/test_image ./ && mv vietnamese/train_
↪images ./ && mv vietnamese/unseen_test_images ./
rm -rf vietnamese

# Rename files
mv labels annotations && mv test_image test && mv train_images  training && mv␣
↪unseen_test_images  unseen_test
mkdir imgs
mv training imgs/ && mv test imgs/ && mv unseen_test imgs/
```

- Step2: Generate `train_labels.json`, `test_labels.json`, `unseen_test_labels.json`, and crop images using 4 processes with the following command (add `--preserve-vertical` if you wish to preserve the images containing vertical texts).

```
python tools/dataset_converters/textrecog/vintext_converter.py PATH/TO/vietnamese␣
↪--nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── vintext
|   ├── crops
|   ├── ignores
|   ├── train_labels.json
|   ├── test_labels.json
|   └── unseen_test_labels.json
```

24.16 BID

警告: This section is not fully tested yet.

- Step1: Download BID Dataset.zip
- Step2: Run the following commands to preprocess the dataset

```
# Rename
mv BID\ Dataset.zip BID_Dataset.zip

# Unzip and Rename
unzip -q BID_Dataset.zip && rm BID_Dataset.zip
```

(下页继续)

(续上页)

```
mv BID\ Dataset BID

# The BID dataset has a problem of permission, and you may
# add permission for this file
chmod -R 777 BID
cd BID
mkdir imgs && mkdir annotations

# For images and annotations
mv CNH_Aberta/*in.jpg imgs && mv CNH_Aberta/*txt annotations && rm -rf CNH_Aberta
mv CNH_Frente/*in.jpg imgs && mv CNH_Frente/*txt annotations && rm -rf CNH_Frente
mv CNH_Verso/*in.jpg imgs && mv CNH_Verso/*txt annotations && rm -rf CNH_Verso
mv CPF_Frente/*in.jpg imgs && mv CPF_Frente/*txt annotations && rm -rf CPF_Frente
mv CPF_Verso/*in.jpg imgs && mv CPF_Verso/*txt annotations && rm -rf CPF_Verso
mv RG_Aberto/*in.jpg imgs && mv RG_Aberto/*txt annotations && rm -rf RG_Aberto
mv RG_Frente/*in.jpg imgs && mv RG_Frente/*txt annotations && rm -rf RG_Frente
mv RG_Verso/*in.jpg imgs && mv RG_Verso/*txt annotations && rm -rf RG_Verso

# Remove unnecessary files
rm -rf desktop.ini
```

- Step3: Generate `train_labels.json` and `val_label.json` (optional) and crop images using 4 processes with the following command (add `--preserve-vertical` if you wish to preserve the images containing vertical texts). Since the original dataset doesn't have a validation set, you may specify `--val-ratio` to split the dataset. E.g., if test-ratio is 0.2, then 20% of the data are left out as the validation set in this example.

```
python tools/dataset_converters/textrecog/bid_converter.py PATH/TO/BID --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── BID
|   ├── crops
|   ├── ignores
|   ├── train_labels.json
|   └── val_label.json (optional)
```

24.17 RCTW

警告： This section is not fully tested yet.

- Step1: Download `train_images.zip.001`, `train_images.zip.002`, and `train_gts.zip` from the homepage, extract the zips to `rctw/imgs` and `rctw/annotations`, respectively.
- Step2: Generate `train_labels.json` and `val_label.json` (optional). Since the original dataset doesn't have a validation set, you may specify `--val-ratio` to split the dataset. E.g., if val-ratio is 0.2, then 20% of the data are left out as the validation set in this example.

```
# Annotations of RCTW test split is not publicly available, split a validation␣
↪set by adding --val-ratio 0.2
# Add --preserve-vertical to preserve vertical texts for training, otherwise␣
↪vertical images will be filtered and stored in PATH/TO/rctw/ignores
python tools/dataset_converters/textrecog/rctw_converter.py PATH/TO/rctw --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── rctw
│   ├── crops
│   ├── ignores
│   ├── train_labels.json
│   └── val_label.json (optional)
```

24.18 HierText

警告： This section is not fully tested yet.

- Step1 (optional): Install AWS CLI.
- Step2: Clone HierText repo to get annotations

```
mkdir HierText
git clone https://github.com/google-research-datasets/hiertext.git
```

- Step3: Download `train.tgz`, `validation.tgz` from aws

```
aws s3 --no-sign-request cp s3://open-images-dataset/ocr/train.tgz .
aws s3 --no-sign-request cp s3://open-images-dataset/ocr/validation.tgz .
```

- Step4: Process raw data

```
# process annotations
mv hiertext/gt ./
rm -rf hiertext
mv gt annotations
gzip -d annotations/train.json.gz
gzip -d annotations/validation.json.gz
# process images
mkdir imgs
mv train.tgz imgs/
mv validation.tgz imgs/
tar -xzvf imgs/train.tgz
tar -xzvf imgs/validation.tgz
```

- Step5: Generate `train_labels.json` and `val_label.json`. HierText includes different levels of annotation, including `paragraph`, `line`, and `word`. Check the original paper for details. E.g. set `--level paragraph` to get paragraph-level annotation. Set `--level line` to get line-level annotation. set `--level word` to get word-level annotation.

```
# Collect word annotation from HierText  --level word
# Add --preserve-vertical to preserve vertical texts for training, otherwise␣
↪vertical images will be filtered and stored in PATH/TO/HierText/ignores
python tools/dataset_converters/textrecog/hiertext_converter.py PATH/TO/HierText -
↪-level word --nproc 4
```

- After running the above codes, the directory structure should be as follows:

```
├── HierText
│   ├── crops
│   ├── ignores
│   ├── train_labels.json
│   └── val_label.json
```

24.19 ArT

警告: This section is not fully tested yet.

- Step1: Download `train_images.tar.gz`, and `train_labels.json` from the homepage to `art/`

```
mkdir art && cd art
mkdir annotations

# Download ArT dataset
wget https://dataset-bj.cdn.bcebos.com/art/train_task2_images.tar.gz
wget https://dataset-bj.cdn.bcebos.com/art/train_task2_labels.json

# Extract
tar -xf train_task2_images.tar.gz
mv train_task2_images crops
mv train_task2_labels.json annotations/

# Remove unnecessary files
rm train_images.tar.gz
```

- Step2: Generate `train_labels.json` and `val_label.json` (optional). Since the test annotations are not publicly available, you may specify `--val-ratio` to split the dataset. E.g., if val-ratio is 0.2, then 20% of the data are left out as the validation set in this example.

```
# Annotations of ArT test split is not publicly available, split a validation set␣
↪by adding --val-ratio 0.2
python tools/dataset_converters/textrecog/art_converter.py PATH/TO/art
```

- After running the above codes, the directory structure should be as follows:

```
|── art
|   ├── crops
|   ├── train_labels.json
|   └── val_label.json (optional)
```

CHAPTER 25

关键信息提取

注解： 我们正努力往*Dataset Preparer* 中增加更多数据集。对于*Dataset Preparer* 暂未能完整支持的数据集，本页提供了一系列手动下载的步骤，供有需要的用户使用。

25.1 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
```

25.2 准备步骤

25.2.1 WildReceipt

- 下载并解压 wildreceipt.tar

25.2.2 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 26

总览

26.1 权重

以下是可用于推理的权重列表。

为了便于使用，有的权重可能会存在多个较短的别名，这在表格中将用“/”分隔。

例如，表格中展示的 DB_r18 / dbnet_resnet18_fpnc_1200e_icdar2015 表示您可以使用 DB_r18 或 dbnet_resnet18_fpnc_1200e_icdar2015 来初始化推理器：

```
>>> from mmocr.apis import TextDetInferencer
>>> inferencer = TextDetInferencer(model='DB_r18')
>>> # 等价于
>>> inferencer = TextDetInferencer(model='dbnet_resnet18_fpnc_1200e_icdar2015')
```

26.1.1 文字检测

模型	README	ICDAR2015 (hmean-iou)	CTW1500 (hmean-iou)	Totaltext (hmean-iou)
DB_r18/dbnet_resnet18_fpnc_1200e_icdar2015	链接	0.8169	-	-
dbnet_resnet50_fpnc_1200e_icdar2015	链接	0.8504	-	-
dbnet_resnet50-dcnv2_fpnc_1200e_icdar2015	链接	0.8543	-	-
DB_r50 / DBNet / dbnet_resnet50-oclip_fpnc_1200e_icdar2015	链接	0.8644	-	-
dbnet_resnet18_fpnc_1200e_totaltext	链接	-	-	0.8182
DBPP_r50/dbnetpp_resnet50_fpnc_1200e_icdar2015	链接	0.8632	-	-
dbnetpp_resnet50-dcnv2_fpnc_1200e_icdar2015	链接	0.8684	-	-
DBNetpp/dbnetpp_resnet50-oclip_fpnc_1200e_icdar2015	链接	0.8882	-	-
MaskRCNN_CTW / mask-rcnn_resnet50_fpn_160e_ctw1500	链接	-	0.7458	-
mask-rcnn_resnet50-oclip_fpn_160e_ctw1500	链接	-	0.7562	-
MaskRCNN_IC15 / mask-rcnn_resnet50_fpn_160e_icdar2015	链接	0.8182	-	-
MaskRCNN/mask-rcnn_resnet50-oclip_fpn_160e_icdar2015	链接	0.8513	-	-
DRRG/drrg_resnet50_fpn-unet_1200e_ctw1500	链接	-	0.8467	-
FCE_CTW_DCNv2 / fcenet_resnet50-dcnv2_fpn_1500e_ctw1500	链接	-	0.8488	-
fcenet_resnet50-oclip_fpn_1500e_ctw1500	链接	-	0.8192	-
FCE_IC15/fcenet_resnet50_fpn_1500e_icdar2015	链接	0.8528	-	-
FCENet/fcenet_resnet50-oclip_fpn_1500e_icdar2015	链接	0.8604	-	-
fcenet_resnet50_fpn_1500e_totaltext	链接	-	-	0.8134
PANet_CTW/panet_resnet18_fpem-ffm_600e_ctw1500	链接	-	0.777	-
PANet_IC15/panet_resnet18_fpem-ffm_600e_icdar2015	链接	0.7848	-	-
PS_CTW/psenet_resnet50_fpnf_600e_ctw1500	链接	-	0.7793	-
psenet_resnet50-oclip_fpnf_600e_ctw1500	链接	-	0.8037	-
PS_IC15/psenet_resnet50_fpnf_600e_icdar2015	链接	0.7998	-	-
PSENet/psenet_resnet50-oclip_fpnf_600e_icdar2015	链接	0.8478	-	-
textsnake_resnet50_fpn-unet_1200e_ctw1500	链接	-	0.8286	-
TextSnake/textsnake_resnet50-oclip_fpn-unet_1200e_ctw1500	链接	-	0.8529	-

26.1.2 文字识别

注解： Avg 指该模型在 IIIT5K、SVT、ICDAR2013、ICDAR2015、SVTP、CT80 上的平均结果。

模型	README	Avg (word_acc)	IIIT5K (word_acc)	SVT (word_acc)	IC-DAR2013 (word_acc)	IC-DAR2015 (word_acc)	SVTP (word_acc)	CT80 (word_acc)
ABINet_Vision / abinet-vision_20e_st-an_mj	链接	0.88	0.95	0.91	0.94	0.79	0.84	0.84
ABINet / abinet_20e_st-an_mj	链接	0.91	0.96	0.94	0.95	0.81	0.89	0.88
ASTER / aster_resnet45_6e_st_mj	链接	0.86	0.94	0.89	0.93	0.77	0.81	0.85
CRNN / crnn_mini-vgg_5e_mj	链接	0.70	0.81	0.81	0.87	0.56	0.61	0.57
MASTER / master_resnet31_12e_st_mj_sa	链接	0.88	0.95	0.90	0.95	0.76	0.85	0.89
nrtr_modality-transform_6e_st_mj	链接	0.83	0.92	0.88	0.94	0.72	0.78	0.75
NRTR / NRTR_1/8-1/4 / nrtr_resnet31-1by8-1by4_6e_st_mj	链接	0.87	0.95	0.88	0.95	0.76	0.80	0.89
NRTR_1/16-1/8 / nrtr_resnet31-1by16-1by8_6e_st_mj	链接	0.87	0.95	0.90	0.94	0.74	0.80	0.89
svtr-small / svtr-small_20e_st_mj	链接	0.86	0.86	0.90	0.94	0.75	0.85	0.89
svtr-base / svtr-base_20e_st_mj	链接	0.87	0.86	0.92	0.94	0.74	0.84	0.90
RobustScanner / robustscanner_resnet31_5e_st-sub_mj-sub_sa_real	链接	0.87	0.95	0.89	0.93	0.76	0.81	0.87
SAR / sar_resnet31_parallel-decoder_5e_st-sub_mj-sub_sa_real	链接	0.88	0.95	0.88	0.94	0.76	0.83	0.90
sar_resnet31_sequential-decoder_5e_st-sub_mj-sub_sa_real	链接	0.87	0.96	0.87	0.94	0.77	0.81	0.89
SATRN / satrn_shallow_5e_st_mj	链接	0.90	0.96	0.92	0.96	0.80	0.88	0.90
SATRN_sm / satrn_shallow-small_5e_st_mj	链接	0.88	0.94	0.90	0.96	0.79	0.86	0.85

26.1.3 关键信息提取

模型	README	wildreceipt (macro_f1)
SDMGR / sdmgr_unet16_60e_wildreceipt	链接	0.89
sdmgr_novisual_60e_wildreceipt	链接	0.87
sdmgr_novisual_60e_wildreceipt_openset	链接	0.93

26.2 统计数据

- 模型权重文件数量：48
- 配置文件数量：49
- 论文数量：19
  - ALGORITHM: 19

26.3 骨干网络

- 模型权重文件数量：1
- 配置文件数量：0
- 论文数量：1
  - [ALGORITHM] *Language Matters: A Weakly Supervised Vision-Language Pre-Training Approach for Scene Text Detection and Spotting*

26.4 文本检测模型

- 模型权重文件数量：29
- 配置文件数量：29
- 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*

26.5 文本识别模型

- 模型权重文件数量：16
- 配置文件数量：17
- 论文数量：9
  - [ALGORITHM] *An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition*
  - [ALGORITHM] *Aster: An Attentional Scene Text Recognizer With Flexible Rectification*
  - [ALGORITHM] *Master: Multi-Aspect Non-Local Network for 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] *Svtr: Scene Text Recognition With a Single Visual Model*

26.6 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*

CHAPTER 27

前沿模型

这里是一些已经复现，但是尚未包含在 MMOCR 包中的前沿模型。

27.1 ABCNet: Real-time Scene Text Spotting with Adaptive Bezier-Curve Network

This is an implementation of ABCNet based on MMOCR, MMCV, and MMEngine.

ABCNet is a conceptually novel, efficient, and fully convolutional framework for text spotting, which address the problem by proposing the Adaptive Bezier-Curve Network (ABCNet). Our contributions are three-fold: 1) For the first time, we adaptively fit arbitrarily-shaped text by a parameterized Bezier curve. 2) We design a novel BezierAlign layer for extracting accurate convolution features of a text instance with arbitrary shapes, significantly improving the precision compared with previous methods. 3) Compared with standard bounding box detection, our Bezier curve detection introduces negligible computation overhead, resulting in superiority of our method in both efficiency and accuracy. Experiments on arbitrarily-shaped benchmark datasets, namely Total-Text and CTW1500, demonstrate that ABCNet achieves state-of-the-art accuracy, meanwhile significantly improving the speed. In particular, on Total-Text, our realtime version is over 10 times faster than recent state-of-the-art methods with a competitive recognition accuracy.

27.1.1 模型状态

推理	训练	README
✓	✓	link

27.2 ABCNet v2: Adaptive Bezier-Curve Network for Real-time End-to-end Text Spotting

This is an implementation of ABCNetV2 based on MMOCR, MMCV, and MMEngine.

ABCNetV2 contributions are four-fold: 1) For the first time, we adaptively fit arbitrarily-shaped text by a parameterized Bezier curve, which, compared with segmentation-based methods, can not only provide structured output but also controllable representation. 2) We design a novel BezierAlign layer for extracting accurate convolution features of a text instance of arbitrary shapes, significantly improving the precision of recognition over previous methods. 3) Different from previous methods, which often suffer from complex post-processing and sensitive hyper-parameters, our ABCNet v2 maintains a simple pipeline with the only post-processing non-maximum suppression (NMS). 4) As the performance of text recognition closely depends on feature alignment, ABCNet v2 further adopts a simple yet effective coordinate convolution to encode the position of the convolutional filters, which leads to a considerable improvement with negligible computation overhead. Comprehensive experiments conducted on various bilingual (English and Chinese) benchmark datasets demonstrate that ABCNet v2 can achieve state-of-the-art performance while maintaining very high efficiency.

27.2.1 模型状态

推理	训练	README
✓	✗	link

27.3 SPTS: Single-Point Text Spotting

This is an implementation of SPTS based on MMOCR, MMCV, and MMEngine.

Existing scene text spotting (i.e., end-to-end text detection and recognition) methods rely on costly bounding box annotations (e.g., text-line, word-level, or character-level bounding boxes). For the first time, we demonstrate that training scene text spotting models can be achieved with an extremely low-cost annotation of a single-point for each instance. We propose an end-to-end scene text spotting method that tackles scene text spotting as a sequence prediction task. Given an image as input, we formulate the desired detection and recognition results as a sequence of discrete tokens and use an auto-regressive Transformer to predict the sequence. The proposed method is simple yet effective, which can achieve state-of-the-art results on widely used benchmarks. Most significantly, we show that the performance is not very sensitive

to the positions of the point annotation, meaning that it can be much easier to be annotated or even be automatically generated than the bounding box that requires precise positions. We believe that such a pioneer attempt indicates a significant opportunity for scene text spotting applications of a much larger scale than previously possible.

27.3.1 模型状态

推理	训练	README
✓	✓	link

CHAPTER 28

骨干网络

28.1 oCLIP

Language Matters: A Weakly Supervised Vision-Language Pre-training Approach for Scene Text Detection and Spotting

28.1.1 Abstract

Recently, Vision-Language Pre-training (VLP) techniques have greatly benefited various vision-language tasks by jointly learning visual and textual representations, which intuitively helps in Optical Character Recognition (OCR) tasks due to the rich visual and textual information in scene text images. However, these methods cannot well cope with OCR tasks because of the difficulty in both instance-level text encoding and image-text pair acquisition (i.e. images and captured texts in them). This paper presents a weakly supervised pre-training method, oCLIP, which can acquire effective scene text representations by jointly learning and aligning visual and textual information. Our network consists of an image encoder and a character-aware text encoder that extract visual and textual features, respectively, as well as a visual-textual decoder that models the interaction among textual and visual features for learning effective scene text representations. With the learning of textual features, the pre-trained model can attend texts in images well with character awareness. Besides, these designs enable the learning from weakly annotated texts (i.e. partial texts in images without text bounding boxes) which mitigates the data annotation constraint greatly. Experiments over the weakly annotated images in ICDAR2019-LSVT show that our pre-trained model improves F-score by +2.5% and +4.8% while transferring its weights to other text detection and spotting networks, respectively. In addition, the proposed method outperforms existing pre-training techniques consistently across multiple public datasets (e.g., +3.2% and +1.3% for Total-Text and CTW1500).

28.1.2 Models

注解： The model is converted from the official oCLIP.

28.1.3 Supported Text Detection Models

28.1.4 Citation

```
@article{xue2022language,
  title={Language Matters: A Weakly Supervised Vision-Language Pre-training Approach for Scene Text Detection and Spotting},
  author={Xue, Chuhui and Zhang, Wenqing and Hao, Yu and Lu, Shijian and Torr, Philip and Bai, Song},
  journal={Proceedings of the European Conference on Computer Vision (ECCV)},
  year={2022}
}
```

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

SynthText

ICDAR2015

Total Text

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

SynthText

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},
  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

Total Text

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

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 ASTER

ASTER: An Attentional Scene Text Recognizer with Flexible Rectification

30.2.1 Abstract

A challenging aspect of scene text recognition is to handle text with distortions or irregular layout. In particular, perspective text and curved text are common in natural scenes and are difficult to recognize. In this work, we introduce ASTER, an end-to-end neural network model that comprises a rectification network and a recognition network. The rectification network adaptively transforms an input image into a new one, rectifying the text in it. It is powered by a flexible Thin-Plate Spline transformation which handles a variety of text irregularities and is trained without human annotations. The recognition network is an attentional sequence-to-sequence model that predicts a character sequence directly from the rectified image. The whole model is trained end to end, requiring only images and their groundtruth text. Through extensive experiments, we verify the effectiveness of the rectification and demonstrate the state-of-the-art recognition performance of ASTER. Furthermore, we demonstrate that ASTER is a powerful component in end-to-end recognition systems, for its ability to enhance the detector.

30.2.2 Dataset

Train Dataset

Test Dataset

30.2.3 Results and models

30.2.4 Citation

```
@article{shi2018aster,
  title={Aster: An attentional scene text recognizer with flexible rectification},
  author={Shi, Baoguang and Yang, Mingkun and Wang, Xinggang and Lyu, Pengyuan and
↪Yao, Cong and Bai, Xiang},
  journal={IEEE transactions on pattern analysis and machine intelligence},
  volume={41},
  number={9},
  pages={2035--2048},
  year={2018},
  publisher={IEEE}
}
```

30.3 CRNN

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

30.3.1 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.

30.3.2 Dataset

Train Dataset

Test Dataset

30.3.3 Results and models

30.3.4 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}
}
```

30.4 MASTER

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

30.4.1 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.

30.4.2 Dataset

Train Dataset

Test Dataset

30.4.3 Results and Models

30.4.4 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}
}
```

30.5 NRTR

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

30.5.1 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).

30.5.2 Dataset

Train Dataset

Test Dataset

30.5.3 Results and Models

30.5.4 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}
}
```

30.6 RobustScanner

RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

30.6.1 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.

30.6.2 Dataset

Train Dataset

Test Dataset

30.6.3 Results and Models

30.6.4 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.

30.6.5 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}
}
```

30.7 SAR

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

30.7.1 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.

30.7.2 Dataset

Train Dataset

Test Dataset

30.7.3 Results and Models

30.7.4 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},
```

(下页继续)

(续上页)

```
  volume={33},
  number={01},
  pages={8610--8617},
  year={2019}
}
```

30.8 SATRN

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

30.8.1 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.

30.8.2 Dataset

Train Dataset

Test Dataset

30.8.3 Results and Models

30.8.4 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}
}
```

30.9 SVTR

SVTR: Scene Text Recognition with a Single Visual Model

30.9.1 Abstract

Dominant scene text recognition models commonly contain two building blocks, a visual model for feature extraction and a sequence model for text transcription. This hybrid architecture, although accurate, is complex and less efficient. In this study, we propose a Single Visual model for Scene Text recognition within the patch-wise image tokenization framework, which dispenses with the sequential modeling entirely. The method, termed SVTR, firstly decomposes an image text into small patches named character components. Afterward, hierarchical stages are recurrently carried out by component-level mixing, merging and/or combining. Global and local mixing blocks are devised to perceive the inter-character and intra-character patterns, leading to a multi-grained character component perception. Thus, characters are recognized by a simple linear prediction. Experimental results on both English and Chinese scene text recognition tasks demonstrate the effectiveness of SVTR. SVTR-L (Large) achieves highly competitive accuracy in English and outperforms existing methods by a large margin in Chinese, while running faster. In addition, SVTR-T (Tiny) is an effective and much smaller model, which shows appealing speed at inference.

30.9.2 Dataset

Train Dataset

Test Dataset

30.9.3 Results and Models

注解: The implementation and configuration follow the original code and paper, but there is still a gap between the reproduced results and the official ones. We appreciate any suggestions to improve its performance.

30.9.4 Citation

```
@inproceedings{ijcai2022p124,
  title     = {SVTR: Scene Text Recognition with a Single Visual Model},
  author    = {Du, Yongkun and Chen, Zhineng and Jia, Caiyan and Yin, Xiaoting and
↪Zheng, Tianlun and Li, Chenxia and Du, Yuning and Jiang, Yu-Gang},
  booktitle = {Proceedings of the Thirty-First International Joint Conference on
               Artificial Intelligence, {IJCAI-22}},
  publisher = {International Joint Conferences on Artificial Intelligence
↪Organization},
```

(下页继续)

(续上页)

```
  editor    = {Lud De Raedt},
  pages     = {884--890},
  year      = {2022},
  month     = {7},
  note      = {Main Track},
  doi       = {10.24963/ijcai.2022/124},
  url       = {https://doi.org/10.24963/ijcai.2022/124},
}
```

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

分支

本文档旨在全面解释 MMOCR 中每个分支的目的和功能。

32.1 分支概述

32.1.1 1. `main`

`main` 分支是 MMOCR 项目的默认分支。它包含了 MMOCR 的最新稳定版本，目前包含了 MMOCR 1.x（例如 v1.0.0）的代码。`main` 分支确保用户能够使用最新和最可靠的软件版本。

32.1.2 2. `dev-1.x`

`dev-1.x` 分支用于开发 MMOCR 的下一个版本。此分支将在发版前进行依赖性测试，通过的提交将会合成到新版本中，并被发布到 `main` 分支。通过设置单独的开发分支，项目可以在不影响 `main` 分支稳定性的情况下继续发展。**所有 PR 应合并到 `dev-1.x` 分支**。

32.1.3 3. 0.x

0.x 分支用作 MMOCR 0.x（例如 v0.6.3）的存档。此分支将不再积极接受更新或改进，但它仍可作为历史参考，或供尚未升级到 MMOCR 1.x 的用户使用。

32.1.4 4. 1.x

它是 main 分支的别名，旨在实现从兼容性时期平稳过渡。它将在 2023 年的年中删除。

注解： 分支映射在 2023.04.06 发生了变化。有关旧分支映射和迁移指南，请参阅分支迁移指南。

CHAPTER 33

贡献指南

OpenMMLab 欢迎所有人参与我们项目的共建。本文档将指导您如何通过拉取请求为 OpenMMLab 项目作出贡献。

33.1 什么是拉取请求？

拉取请求 (Pull Request), GitHub 官方文档定义如下。

> 拉取请求是一种通知机制。你修改了他人的代码，将你的修改通知原来作者，希望他合并你的修改。

33.2 基本的工作流：

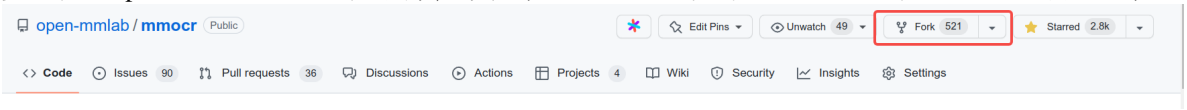
1. 获取最新的代码库
2. 从最新的 `dev-1.x` 分支创建分支进行开发
3. 提交修改 (*不要忘记使用 pre-commit hooks!*)
4. 推送你的修改并创建一个 拉取请求
5. 讨论、审核代码
6. 将开发分支合并到 `dev-1.x` 分支

33.3 具体步骤

33.3.1 1. 获取最新的代码库

- 当你第一次提 PR 时

  复刻 OpenMMLab 原代码库，点击 GitHub 页面右上角的 **Fork** 按钮即可



  克隆复刻的代码库到本地

```
git clone git@github.com:XXX/mmocr.git
```

  添加原代码库为上游代码库

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

- 从第二个 PR 起

  检出本地代码库的主分支，然后从最新的原代码库的主分支拉取更新。这里假设你正基于 `dev-1.x` 开发。

```
git checkout dev-1.x
git pull upstream dev-1.x
```

33.3.2 2. 从 `dev-1.x` 分支创建一个新的开发分支

```
git checkout -b branchname
```

小技巧： 为了保证提交历史清晰可读，我们强烈推荐您先切换到 `dev-1.x` 分支，再创建新的分支。

33.3.3 3. 提交你的修改

- 如果你是第一次尝试贡献，请在 MMOCR 的目录下安装并初始化 pre-commit hooks。

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

- 提交修改。在每次提交前，pre-commit hooks 都会被触发并规范化你的代码格式。

```
# coding
git add [files]
git commit -m 'messages'
```

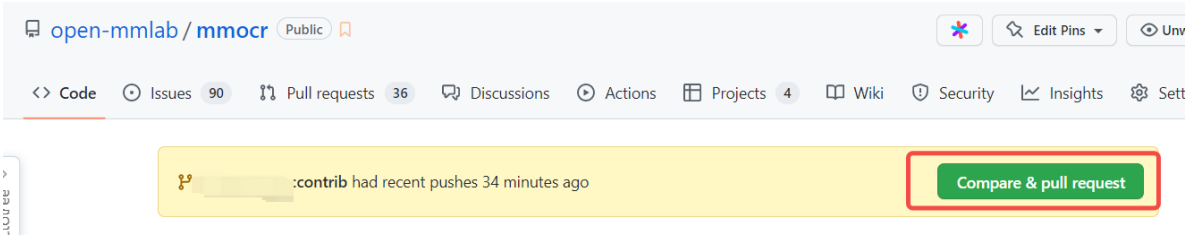
注解： 有时你的文件可能会在提交时被 pre-commit hooks 自动修改。这时请重新添加并提交修改后的文件。

33.3.4 4. 推送你的修改到复刻的代码库，并创建一个拉取请求

- 推送当前分支到远端复刻的代码库

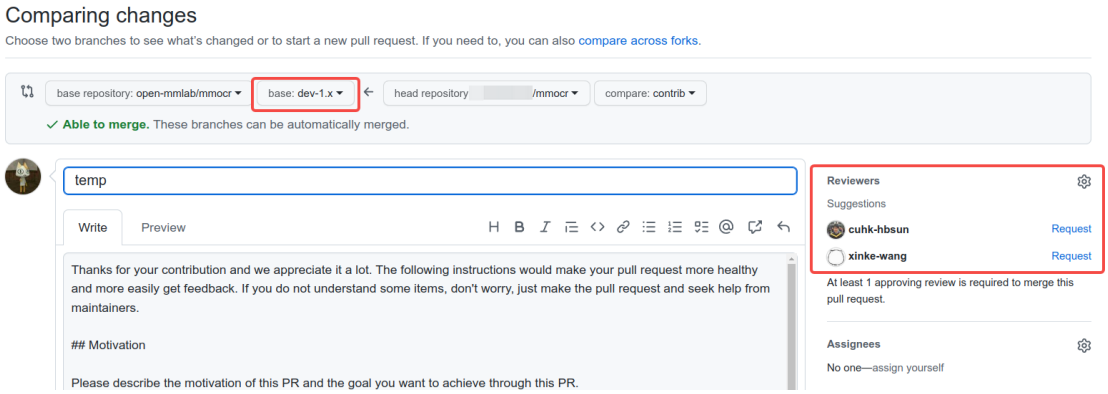
```
git push origin branchname
```

- 创建一个拉取请求



- 修改拉取请求信息模板，描述修改原因和修改内容。还可以在 PR 描述中，手动关联到相关的议题 (issue)，(更多细节，请参考官方文档)。

- 另外，如果你正在往 `dev-1.x` 分支提交代码，你还需要在创建 PR 的界面中将基础分支改为 `dev-1.x`，因为现在默认的基础分支是 `main`。



- 你同样可以把 PR 关联给相关人员进行评审。

33.3.5 5. 讨论并评审你的代码

- 根据评审人员的意见修改代码，并推送修改

33.3.6 6. 拉取请求合并之后删除该分支

- 在 PR 合并之后，你就可以删除该分支了。

```
git branch -d branchname # 删除本地分支
git push origin --delete branchname # 删除远程分支
```

33.4 PR 规范

1. 使用 pre-commit hook，尽量减少代码风格相关问题
2. 一个 PR 对应一个短期分支
3. 粒度要细，一个 PR 只做一件事情，避免超大的 PR
   - Bad：实现 Faster R-CNN
   - Acceptable：给 Faster R-CNN 添加一个 box head
   - Good：给 box head 增加一个参数来支持自定义的 conv 层数
4. 每次 Commit 时需要提供清晰且有意义 commit 信息
5. 提供清晰且有意义的拉取请求描述
   - 标题写明白任务名称，一般格式:[Prefix] Short description of the pull request (Suffix)
   - prefix: 新增功能 [Feature], 修 bug [Fix], 文档相关 [Docs], 开发中 [WIP] (暂时不会被 review)
   - 描述里介绍拉取请求的主要修改内容，结果，以及对其他部分的影响, 参考拉取请求模板
   - 关联相关的议题 (issue) 和其他拉取请求

CHAPTER 34

Changelog of v1.x

34.1 v1.0.0 (04/06/2023)

We are excited to announce the first official release of MMOCR 1.0, with numerous enhancements, bug fixes, and the introduction of new dataset support!

34.1.1 Highlights

- Support for SCUT-CTW1500, SynthText, and MJSynth datasets
- Updated FAQ and documentation
- Deprecation of file_client_args in favor of backend_args
- Added a new MMOCR tutorial notebook

34.1.2 New Features & Enhancement

- Add SCUT-CTW1500 by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1677
- Cherry Pick #1205 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1774
- Make lanms-neo optional by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1772
- SynthText by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1779

- Deprecate file_client_args and use backend_args instead by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1765
- MJSynth by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1791
- Add MMOCR tutorial notebook by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1771
- decouple batch_size to det_batch_size, rec_batch_size and kie_batch_size in MMOCRInferencer by @hugotong6425 in https://github.com/open-mmlab/mmocr/pull/1801
- Accepts local-rank in train.py and test.py by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1806
- update stitch_boxes_into_lines by @cherryjm in https://github.com/open-mmlab/mmocr/pull/1824
- Add tests for pytorch 2.0 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1836

34.1.3 Docs

- FAQ by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1773
- Remove LoadImageFromLMDB from docs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1767
- Mark projects in docs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1766
- add opendatalab download link by @jorie-peng in https://github.com/open-mmlab/mmocr/pull/1753
- Fix some deadlinks in the docs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1469
- Fix quick run by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1775
- Dataset by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1782
- Update faq by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1817
- more social network links by @fengshiwest in https://github.com/open-mmlab/mmocr/pull/1818
- Update docs after branch switching by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1834

34.1.4 Bug Fixes:

- Place dicts to .mim by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1781
- Test svtr_small instead of svtr_tiny by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1786
- Add pse weight to metafile by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1787
- Synthtext metafile by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1788
- Clear up some unused scripts by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1798
- if dst not exists, when move a single file may raise a file not exists error. by @KevinNuNu in https://github.com/open-mmlab/mmocr/pull/1803

- CTW1500 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1814
- MJSynth & SynthText Dataset Preparer config by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1805
- Use poly_intersection instead of poly.intersection to avoid sup…by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1811
- Abinet: fix ValueError: Blur limit must be odd when centered=True. Got: (3, 6) by @hugotong6425 in https://github.com/open-mmlab/mmocr/pull/1821
- Bug generated during kie inference visualization by @Yangget in https://github.com/open-mmlab/mmocr/pull/1830
- Revert sync bn in inferencer by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1832
- Fix mmdet digit version by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1840

34.1.5 New Contributors

- @jorie-peng made their first contribution in https://github.com/open-mmlab/mmocr/pull/1753
- @hugotong6425 made their first contribution in https://github.com/open-mmlab/mmocr/pull/1801
- @fengshiwest made their first contribution in https://github.com/open-mmlab/mmocr/pull/1818
- @cherryjm made their first contribution in https://github.com/open-mmlab/mmocr/pull/1824
- @Yangget made their first contribution in https://github.com/open-mmlab/mmocr/pull/1830

Thank you to all the contributors for making this release possible! We're excited about the new features and enhancements in this version, and we're looking forward to your feedback and continued support. Happy coding!

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v1.0.0rc6…v1.0.0

34.1.6 Highlights

34.2 v1.0.0rc6 (03/07/2023)

34.2.1 Highlights

1. Two new models, ABCNet v2 (inference only) and SPTS are added to `projects/` folder.
2. Announcing `Inferencer`, a unified inference interface in OpenMMLab for everyone's easy access and quick inference with all the pre-trained weights. Docs
3. Users can use test-time augmentation for text recognition tasks. Docs
4. Support batch augmentation through `BatchAugSampler`, which is a technique used in SPTS.

5. Dataset Preparer has been refactored to allow more flexible configurations. Besides, users are now able to prepare text recognition datasets in LMDB formats. Docs
6. Some textspotting datasets have been revised to enhance the correctness and consistency with the common practice.
7. Potential spurious warnings from `shapely` have been eliminated.

34.2.2 Dependency

This version requires MMEngine >= 0.6.0, MMCV >= 2.0.0rc4 and MMDet >= 3.0.0rc5.

34.2.3 New Features & Enhancements

- Discard deprecated lmdb dataset format and only support img+label now by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1681
- abcnetv2 inference by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1657
- Add RepeatAugSampler by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1678
- SPTS by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1696
- Refactor Inferencers by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1608
- Dynamic return type for rescale_polygons by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1702
- Revise upstream version limit by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1703
- TextRecogCropConverter add crop with opencv warpPersepective function by @KevinNuNu in https://github.com/open-mmlab/mmocr/pull/1667
- change cudnn benchmark to false by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1705
- Add ST-pretrained DB-series models and logs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1635
- Only keep meta and state_dict when publish model by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1729
- Rec TTA by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1401
- Speedup formatting by replacing np.transpose with torch⋯by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1719
- Support auto import modules from registry. by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1731
- Support batch visualization & dumping in Inferencer by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1722
- add a new argument font_properties to set a specific font file in order to draw Chinese characters properly by @KevinNuNu in https://github.com/open-mmlab/mmocr/pull/1709

- Refactor data converter and gather by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1707
- Support batch augmentation through BatchAugSampler by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1757
- Put all registry into registry.py by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1760
- train by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1756
- configs for regression benchmark by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1755
- Support lmdb format in Dataset Preparer by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1762

34.2.4 Docs

- update the link of DBNet by @AllentDan in https://github.com/open-mmlab/mmocr/pull/1672
- Add notice for default branch switching by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1693
- docs: Add twitter discord medium youtube link by @vansin in https://github.com/open-mmlab/mmocr/pull/1724
- Remove unsupported datasets in docs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1670

34.2.5 Bug Fixes

- Update dockerfile by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1671
- Explicitly create np object array for compatibility by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1691
- Fix a minor error in docstring by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1685
- Fix lint by @triple-Mu in https://github.com/open-mmlab/mmocr/pull/1694
- Fix LoadOCRAnnotation ut by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1695
- Fix isort pre-commit error by @KevinNuNu in https://github.com/open-mmlab/mmocr/pull/1697
- Update owners by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1699
- Detect intersection before using shapley.intersection to eliminate spurious warnings by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1710
- Fix some inferencer bugs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1706
- Fix textocr ignore flag by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1712
- Add missing softmax in ASTER forward_test by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1718
- Fix head in readme by @vansin in https://github.com/open-mmlab/mmocr/pull/1727

- Fix some browse dataset script bugs and draw textdet gt instance with ignore flags by @KevinNuNu in https://github.com/open-mmlab/mmocr/pull/1701
- icdar textrecog ann parser skip data with ignore flag by @KevinNuNu in https://github.com/open-mmlab/mmocr/pull/1708
- bezier_to_polygon -> bezier2polygon by @double22a in https://github.com/open-mmlab/mmocr/pull/1739
- Fix docs recog CharMetric P/R error definition by @KevinNuNu in https://github.com/open-mmlab/mmocr/pull/1740
- Remove outdated resources in demo/ by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1747
- Fix wrong ic13 textspotting split data; add lexicons to ic13, ic15 and totaltext by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1758
- SPTS readme by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1761

34.2.6 New Contributors

- @triple-Mu made their first contribution in https://github.com/open-mmlab/mmocr/pull/1694
- @double22a made their first contribution in https://github.com/open-mmlab/mmocr/pull/1739

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v1.0.0rc5…v1.0.0rc6

34.3 v1.0.0rc5 (01/06/2023)

34.3.1 Highlights

1. Two models, Aster and SVTR, are added to our model zoo. The full implementation of ABCNet is also available now.
2. Dataset Preparer supports 5 more datasets: CocoTextV2, FUNSD, TextOCR, NAF, SROIE.
3. We have 4 more text recognition transforms, and two helper transforms. See https://github.com/open-mmlab/mmocr/pull/1646 https://github.com/open-mmlab/mmocr/pull/1632 https://github.com/open-mmlab/mmocr/pull/1645 for details.
4. The transform, `FixInvalidPolygon`, is getting smarter at dealing with invalid polygons, and now capable of handling more weird annotations. As a result, a complete training cycle on TotalText dataset can be performed bug-free. The weights of DBNet and FCENet pretrained on TotalText are also released.

34.3.2 New Features & Enhancements

- Update ic15 det config according to DataPrepare by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1617
- Refactor icdardataset metainfo to lowercase. by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1620
- Add ASTER Encoder by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1239
- Add ASTER decoder by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1625
- Add ASTER config by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1238
- Update ASTER config by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1629
- Support browse_dataset.py to visualize original dataset by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1503
- Add CocoTextv2 to dataset preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1514
- Add Funsd to dataset preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1550
- Add TextOCR to Dataset Preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1543
- Refine example projects and readme by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1628
- Enhance FixInvalidPolygon, add RemoveIgnored transform by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1632
- ConditionApply by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1646
- Add NAF to dataset preparer by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1609
- Add SROIE to dataset preparer by @FerryHuang in https://github.com/open-mmlab/mmocr/pull/1639
- Add svtr decoder by @willpat1213 in https://github.com/open-mmlab/mmocr/pull/1448
- Add missing unit tests by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1651
- Add svtr encoder by @willpat1213 in https://github.com/open-mmlab/mmocr/pull/1483
- ABCNet train by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1610
- Totaltext cfgs for DB and FCE by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1633
- Add Aliases to models by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1611
- SVTR transforms by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1645
- Add SVTR framework and configs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1621
- Issue Template by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1663

34.3.3 Docs

- Add Chinese translation for browse_dataset.py by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1647
- updata abcnet doc by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1658
- update the dbnetpp's readme file by @zhuyue66 in https://github.com/open-mmlab/mmocr/pull/1626
- Inferencer docs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1744

34.3.4 Bug Fixes

- nn.SmoothL1Loss beta can not be zero in PyTorch 1.13 version by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1616
- ctc loss bug if target is empty by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1618
- Add torch 1.13 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1619
- Remove outdated tutorial link by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1627
- Dev 1.x some doc mistakes by @KevinNuNu in https://github.com/open-mmlab/mmocr/pull/1630
- Support custom font to visualize some languages (e.g. Korean) by @ProtossDragoon in https://github.com/open-mmlab/mmocr/pull/1567
- db_module_loss, negative number encountered in sqrt by @KevinNuNu in https://github.com/open-mmlab/mmocr/pull/1640
- Use int instead of np.int by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1636
- Remove support for py3.6 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1660

34.3.5 New Contributors

- @zhuyue66 made their first contribution in https://github.com/open-mmlab/mmocr/pull/1626
- @KevinNuNu made their first contribution in https://github.com/open-mmlab/mmocr/pull/1630
- @FerryHuang made their first contribution in https://github.com/open-mmlab/mmocr/pull/1639
- @willpat1213 made their first contribution in https://github.com/open-mmlab/mmocr/pull/1448

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v1.0.0rc4…v1.0.0rc5

34.4 v1.0.0rc4 (12/06/2022)

34.4.1 Highlights

1. Dataset Preparer can automatically generate base dataset configs at the end of the preparation process, and supports 6 more datasets: IIIT5k, CUTE80, ICDAR2013, ICDAR2015, SVT, SVTP.
2. Introducing our `projects/` folder - implementing new models and features into OpenMMLab' s algorithm libraries has long been complained to be troublesome due to the rigorous requirements on code quality, which could hinder the fast iteration of SOTA models and might discourage community members from sharing their latest outcome here. We now introduce `projects/` folder, where some experimental features, frameworks and models can be placed, only needed to satisfy the minimum requirement on the code quality. Everyone is welcome to post their implementation of any great ideas in this folder! We also add the first example project to illustrate what we expect a good project to have (check out the raw content of README.md for more info!).
3. Inside the `projects/` folder, we are releasing the preview version of ABCNet, which is the first implementation of text spotting models in MMOCR. It' s inference-only now, but the full implementation will be available very soon.

34.4.2 New Features & Enhancements

- Add SVT to dataset preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1521
- Polish bbox2poly by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1532
- Add SVTP to dataset preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1523
- Iiit5k converter by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1530
- Add cute80 to dataset preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1522
- Add IC13 preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1531
- Add 'Projects/' folder, and the first example project by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1524
- Rename to {dataset-name}_task_train/test by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1541
- Add print_config.py to the tools by @IncludeMathH in https://github.com/open-mmlab/mmocr/pull/1547
- Add get_md5 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1553
- Add config generator by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1552
- Support IC15_1811 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1556
- Update CT80 config by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1555
- Add config generators to all textdet and textrecog configs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1560

- Refactor TPS by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1240
- Add TextSpottingConfigGenerator by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1561
- Add common typing by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1596
- Update textrecog config and readme by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1597
- Support head loss or postprocessor is None for only infer by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1594
- Textspotting datasample by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1593
- Simplify mono_gather by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1588
- ABCNet v1 infer by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1598

34.4.3 Docs

- Add Chinese Guidance on How to Add New Datasets to Dataset Preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1506
- Update the qq group link by @vansin in https://github.com/open-mmlab/mmocr/pull/1569
- Collapse some sections; update logo url by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1571
- Update dataset preparer (CN) by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1591

34.4.4 Bug Fixes

- Fix two bugs in dataset preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1513
- Register bug of CLIPResNet by @jyshee in https://github.com/open-mmlab/mmocr/pull/1517
- Being more conservative on Dataset Preparer by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1520
- python -m pip upgrade in windows by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1525
- Fix wildreceipt metafile by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1528
- Fix Dataset Preparer Extract by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1527
- Fix ICDARTxtParser by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1529
- Fix Dataset Zoo Script by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1533
- Fix crop without padding and recog metainfo delete unuse info by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1526
- Automatically create nonexistent directory for base configs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1535

- Change mmcv.dump to mmengine.dump by @ProtossDragoon in https://github.com/open-mmlab/mmocr/pull/1540
- mmocr.utils.typing -> mmocr.utils.typing_utils by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1538
- Wildreceipt tests by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1546
- Fix judge exist dir by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1542
- Fix IC13 textdet config by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1563
- Fix IC13 textrecog annotations by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1568
- Auto scale lr by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1584
- Fix icdar data parse for text containing separator by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1587
- Fix textspotting ut by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1599
- Fix TextSpottingConfigGenerator and TextSpottingDataConverter by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1604
- Keep E2E Inferencer output simple by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1559

34.4.5 New Contributors

- @jyshee made their first contribution in https://github.com/open-mmlab/mmocr/pull/1517
- @ProtossDragoon made their first contribution in https://github.com/open-mmlab/mmocr/pull/1540
- @IncludeMathH made their first contribution in https://github.com/open-mmlab/mmocr/pull/1547

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v1.0.0rc3…v1.0.0rc4

34.5 v1.0.0rc3 (11/03/2022)

34.5.1 Highlights

1. We release several pretrained models using oCLIP-ResNet as the backbone, which is a ResNet variant trained with oCLIP and can significantly boost the performance of text detection models.
2. Preparing datasets is troublesome and tedious, especially in OCR domain where multiple datasets are usually required. In order to free our users from laborious work, we designed a Dataset Preparer to help you get a bunch of datasets ready for use, with only **one line of command**! Dataset Preparer is also crafted to consist of a series of reusable modules, each responsible for handling one of the standardized phases throughout the preparation process, shortening the development cycle on supporting new datasets.

34.5.2 New Features & Enhancements

- Add Dataset Preparer by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1484
- support modified resnet structure used in oCLIP by @HannibalAPE in https://github.com/open-mmlab/mmocr/pull/1458
- Add oCLIP configs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1509

34.5.3 Docs

- Update install.md by @rogachevai in https://github.com/open-mmlab/mmocr/pull/1494
- Refine some docs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1455
- Update some dataset preparer related docs by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1502
- oclip readme by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1505

34.5.4 Bug Fixes

- Fix offline_eval error caused by new data flow by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1500

34.5.5 New Contributors

- @rogachevai made their first contribution in https://github.com/open-mmlab/mmocr/pull/1494
- @HannibalAPE made their first contribution in https://github.com/open-mmlab/mmocr/pull/1458

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v1.0.0rc2…v1.0.0rc3

34.6 v1.0.0rc2 (10/14/2022)

This release relaxes the version requirement of `MMEngine` to `>=0.1.0, < 1.0.0`.

34.7 v1.0.0rc1 (10/09/2022)

34.7.1 Highlights

This release fixes a severe bug leading to inaccurate metric report in multi-GPU training. We release the weights for all the text recognition models in MMOCR 1.0 architecture. The inference shorthand for them are also added back to `ocr.py`. Besides, more documentation chapters are available now.

34.7.2 New Features & Enhancements

- Simplify the Mask R-CNN config by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1391
- auto scale lr by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1326
- Update paths to pretrain weights by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1416
- Streamline duplicated split_result in pan_postprocessor by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1418
- Update model links in ocr.py and inference.md by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1431
- Update rec configs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1417
- Visualizer refine by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1411
- Support get flops and parameters in dev-1.x by @vansin in https://github.com/open-mmlab/mmocr/pull/1414

34.7.3 Docs

- intersphinx and api by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1367
- Fix quickrun by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1374
- Fix some docs issues by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1385
- Add Documents for DataElements by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1381
- config english by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1372
- Metrics by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1399
- Add version switcher to menu by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1407
- Data Transforms by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1392
- Fix inference docs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1415
- Fix some docs by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1410
- Add maintenance plan to migration guide by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1413
- Update Recog Models by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1402

34.7.4 Bug Fixes

- clear metric.results only done in main process by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1379
- Fix a bug in MMDetWrapper by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1393
- Fix browse_dataset.py by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1398
- ImgAugWrapper: Do not cilp polygons if not applicable by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1231
- Fix CI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1365
- Fix merge stage test by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1370
- Del CI support for torch 1.5.1 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1371
- Test windows cu111 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1373
- Fix windows CI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1387
- Upgrade pre commit hooks by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1429
- Skip invalid augmented polygons in ImgAugWrapper by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1434

34.7.5 New Contributors

- @vansin made their first contribution in https://github.com/open-mmlab/mmocr/pull/1414

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v1.0.0rc0…v1.0.0rc1

34.8 v1.0.0rc0 (09/01/2022)

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.

34.8.1 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.

34.8.2 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.

34.8.3 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.

34.8.4 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

概览

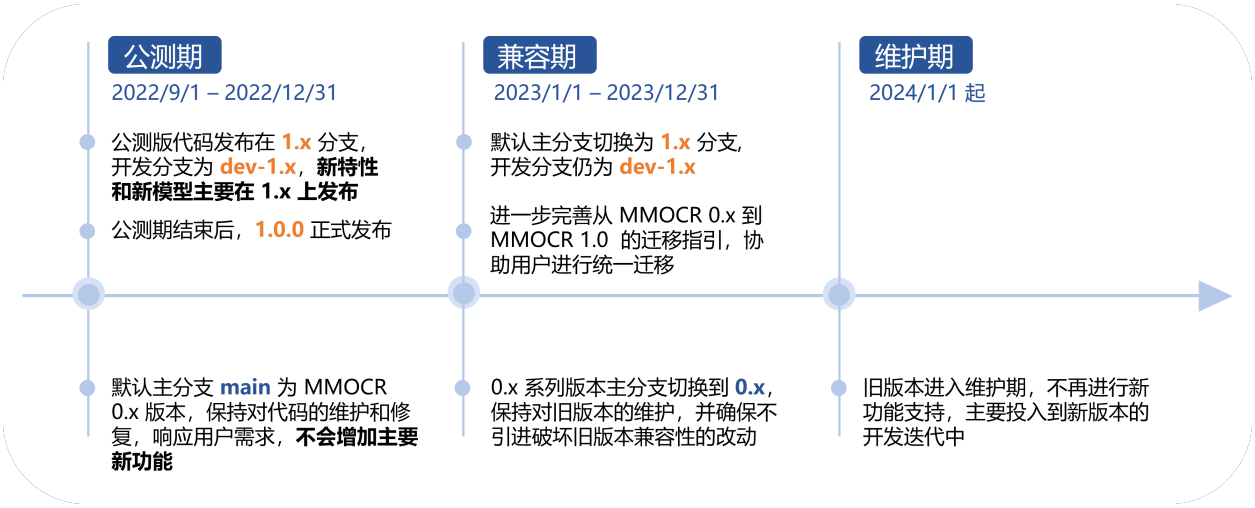
伴随着 OpenMMLab 2.0 的发布，MMOCR 1.0 本身也作出了许多突破性的改变，使得代码的冗余度降低，代码效率提高，整体设计上也变得更为一致。然而，这些改变使得完美的后向兼容不再可能。我们也深知在这样巨大的变动之下，老用户想第一时间适应新版本也绝非易事。因此，我们推出了详细的迁移指南，旨在让老用户们尽可能平滑地过渡到全新的框架，最终能享受到全新的 MMOCR 和整个 OpenMMLab 2.0 生态系统为生产力带来的巨大优势。

警告： MMOCR 1.0 依赖于新的基础训练框架 MMEngine，因而有着与 MMOCR 0.x 完全不同的依赖链。尽管你可能已经拥有了一个可以正常运行 MMOCR 0.x 的环境，但你仍然需要创建一个新的 python 环境来安装 MMOCR 1.0 版本所需要的依赖库。我们提供了详细的安装文档以供参考。

接下来，请根据你的实际需求，阅读需要的章节：

- 若需要了解 MMOCR 1.0 的主要变化，请阅读*MMOCR 1.x 更新汇总*
- 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 ，分支迁移 和 上游依赖库变更

如下图所示，MMOCR 1.x 版本的维护计划主要分为三个阶段，即“公测期”，“兼容期”以及“维护期”。对于旧版本，我们将不再增加主要新功能。因此，我们强烈建议用户尽早迁移至 MMOCR 1.x 版本。


公测期
2022/9/1 – 2022/12/31
公测版代码发布在 1.x 分支，开发分支为 dev-1.x，新特性和新模型主要在 1.x 上发布
公测期结束后，1.0.0 正式发布
默认主分支 main 为 MMOCR 0.x 版本，保持对代码的维护和修复，响应用户需求，不会增加主要新功能
兼容期
2023/1/1 – 2023/12/31
默认主分支切换为 1.x 分支，开发分支仍为 dev-1.x
进一步完善从 MMOCR 0.x 到 MMOCR 1.0 的迁移指引，协助用户进行统一迁移
0.x 系列版本主分支切换到 0.x，保持对旧版本的维护，并确保不引进破坏旧版本兼容性的改动
维护期
2024/1/1 起
旧版本进入维护期，不再进行新功能支持，主要投入到新版本的开发迭代中

CHAPTER 36

MMOCR 1.x 更新汇总

此处列出了 MMOCR 1.x 相对于 0.x 版本的重大更新。

1. 架构升级：MMOCR 1.x 是基于 MMEngine，提供了一个通用的、强大的执行器，允许更灵活的定制，提供了统一的训练和测试入口。
2. 统一接口：MMOCR 1.x 统一了数据集、模型、评估和可视化的接口和内部逻辑。支持更强的扩展性。
3. 跨项目调用：受益于统一的设计，你可以使用其他 OpenMMLab 项目中实现的模型，如 MMDet。我们提供了一个例子，说明如何通过 MMDetWrapper 使用 MMDetection 的 Mask R-CNN。查看我们的文档以了解更多细节。更多的包装器将在未来发布。
4. 更强的可视化：我们提供了一系列可视化工具，用户现在可以更方便可视化数据。
5. 更多的文档和教程：我们增加了更多的教程，降低用户的学习门槛。
6. 一站式数据准备：准备数据集已经不再是难事。使用我们的 Dataset Preparer，一行命令即可让多个数据集准备就绪。
7. 拥抱更多 `projects/`: 我们推出了 `projects/` 文件夹，用于存放一些实验性的新特性、框架和模型。我们对这个文件夹下的代码规范不作过多要求，力求让社区的所有想法第一时间得到实现和展示。请查看我们的样例 project 以了解更多。
8. 更多新模型：MMOCR 1.0 支持了更多模型和模型种类。

CHAPTER 37

分支迁移

在早期阶段，MMOCR 有三个分支：`main`、`1.x` 和 `dev-1.x`。随着 MMOCR 1.0.0 正式版的发布，我们也重命名了其中一些分支，下面提供了新旧分支的对照。

- `main` 分支包括了 MMOCR 0.x（例如 v0.6.3）的代码。现在已经被重命名为 `0.x`。
- `1.x` 包含了 MMOCR 1.x（例如 1.0.0rc6）的代码。现在它是 `main` 分支的别名，会在 2023 的年中删除。
- `dev-1.x` 是 MMOCR 1.x 的开发分支。现在保持不变。

有关分支的更多信息，请查看分支。

37.1 升级 main 分支时解决冲突

对于希望从旧 `main` 分支（包含 MMOCR 0.x 代码）升级的用户，代码可能会导致冲突。要避免这些冲突，请按照以下步骤操作：

1. 请 commit 在 `main` 上的所有更改（若有），并备份您当前的 `main` 分支。

```
git checkout main
git add --all
git commit -m 'backup'
git checkout -b main_backup
```

2. 从远程存储库获取最新更改。

```
git remote add openmmlab git@github.com:open-mmlab/mmocr.git
git fetch openmmlab
```

3. 通过运行 `git reset --hard openmmlab/main` 将 `main` 分支重置为远程存储库上的最新 `main` 分支。

```
git checkout main
git reset --hard openmmlab/main
```

按照这些步骤，您可以成功升级您的 `main` 分支。

CHAPTER 38

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。

38.1 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。

38.2 文本检测

38.2.1 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。

38.2.2 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。

38.2.3 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。

38.2.4 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。

38.2.5 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）。

38.3 文本识别

38.3.1 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。

38.3.2 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。

38.4 关键信息提取

38.4.1 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。

38.4.2 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。

38.4.3 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。

38.5 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 39

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。

39.1 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。

39.1.1 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]]}]}
```

39.1.2 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"}
```

39.2 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`。

39.2.1 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
            },
            // ...
          ]
      }
    ]
}
```

其中，bbox 字段的格式为 [min_x, min_y, max_x, max_y]。

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}
```

39.2.2 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}
```

39.3 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**图像 + 文字**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 imgs.lmdb，该 lmdb 文件**包含标签和图像**。

```
# 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',
    pipeline=None)
```

还需把 train_pipeline 及 test_pipeline 中的数据读取方法如*LoadImageFromFile* 替换为 LoadImageFromNDArray：

```
train_pipeline = [dict(type='LoadImageFromNDArray')]
```

CHAPTER 40

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 41

数据变换迁移

41.1 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。

41.2 配置迁移指南

41.2.1 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'))
```

41.2.2 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 42

mmocr.apis

mmocr.apis

42.1 Inferencers

MMOCRInferencer	MMOCR Inferencer.
TextDetInferencer	Text Detection inferencer.
TextRecInferencer	Text Recognition inferencer.
TextSpotInferencer	Text Spotting inferencer.
KIEInferencer	Key Information Extraction Inferencer.

42.1.1 MMOCRInferencer

class mmocr.apis.inferencers.**MMOCRInferencer**(*det=None*, *det_weights=None*, *rec=None*, *rec_weights=None*, *kie=None*, *kie_weights=None*, *device=None*)

MMOCR Inferencer. It's a wrapper around three base task inferenecers: TextDetInferencer, TextRecInferencer and KIEInferencer, and it can be used to perform end-to-end OCR or KIE inference.

参数

- **det** (*Optional[Union[ConfigType, str]]*) –Pretrained text detection algorithm. It's the path to the config file or the model name defined in metafile. Defaults to None.
- **det_weights** (*Optional[str]*) –Path to the custom checkpoint file of the selected det model. If it is not specified and "det" is a model name of metafile, the weights will be loaded from metafile. Defaults to None.
- **rec** (*Optional[Union[ConfigType, str]]*) –Pretrained text recognition algorithm. It's the path to the config file or the model name defined in metafile. Defaults to None.
- **rec_weights** (*Optional[str]*) –Path to the custom checkpoint file of the selected rec model. If it is not specified and "rec" is a model name of metafile, the weights will be loaded from metafile. Defaults to None.
- **kie** (*Optional[Union[ConfigType, str]]*) –Pretrained key information extraction algorithm. It's the path to the config file or the model name defined in metafile. Defaults to None.
- **kie_weights** (*Optional[str]*) –Path to the custom checkpoint file of the selected kie model. If it is not specified and "kie" is a model name of metafile, the weights will be loaded from metafile. Defaults to None.
- **device** (*Optional[str]*) –Device to run inference. If None, the available device will be automatically used. Defaults to None.

返回类型 None

forward(*inputs*, *batch_size=1*, *det_batch_size=None*, *rec_batch_size=None*, *kie_batch_size=None*, ***forward_kwargs*)

Forward the inputs to the model.

参数

- **inputs** (*InputsType*) –The inputs to be forwarded.
- **batch_size** (*int*) –Batch size. Defaults to 1.
- **det_batch_size** (*Optional[int]*) –Batch size for text detection model. Overwrite batch_size if it is not None. Defaults to None.

- **rec_batch_size** (*Optional[int]*) –Batch size for text recognition model. Overwrite batch_size if it is not None. Defaults to None.
- **kie_batch_size** (*Optional[int]*) –Batch size for KIE model. Overwrite batch_size if it is not None. Defaults to None.

返回 The prediction results. Possibly with keys “det”, “rec”, and “kie”..

返回类型 Dict

postprocess(*preds*, *visualization=None*, *print_result=False*, *save_pred=False*, *pred_out_dir=''*)

Process the predictions and visualization results from `forward` and `visualize`.

This method should be responsible for the following tasks:

1. Convert datasamples into a json-serializable dict if needed.
2. Pack the predictions and visualization results and return them.
3. Dump or log the predictions.

参数

- **preds** (*PredType*) –Predictions of the model.
- **visualization** (*Optional[np.ndarray]*) –Visualized predictions.
- **print_result** (*bool*) –Whether to print the result. Defaults to False.
- **save_pred** (*bool*) –Whether to save the inference result. Defaults to False.
- **pred_out_dir** (*str*) –File to save the inference results w/o visualization. If left as empty, no file will be saved. Defaults to ‘’.

返回

Inference and visualization results, mapped from ”predictions” and “visualization”.

返回类型 Dict

visualize(*inputs*, *preds*, ***kwargs*)

Visualize predictions.

参数

- **inputs** (*List[Union[str, np.ndarray]]*) –Inputs for the inferencer.
- **preds** (*List[Dict]*) –Predictions of the model.
- **show** (*bool*) –Whether to display the image in a popup window. Defaults to False.
- **wait_time** (*float*) –The interval of show (s). Defaults to 0.
- **draw_pred** (*bool*) –Whether to draw predicted bounding boxes. Defaults to True.
- **pred_score_thr** (*float*) –Minimum score of bboxes to draw. Defaults to 0.3.

- **save_vis** (*bool*) –Whether to save the visualization result. Defaults to False.
- **img_out_dir** (*str*) –Output directory of visualization results. If left as empty, no file will be saved. Defaults to ‘’.

返回 Returns visualization results only if applicable.

返回类型 List[np.ndarray] or None

42.1.2 TextDetInferencer

class mmocr.apis.inferencers.**TextDetInferencer**(*model=None*, *weights=None*, *device=None*, *scope='mmocr'*)

Text Detection inferencer.

参数

- **model** (*str, optional*) –Path to the config file or the model name defined in metafile. For example, it could be “dbnet_resnet18_fpnc_1200e_icdar2015” or “configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py”. If model is not specified, user must provide the *weights* saved by MMEngine which contains the config string. Defaults to None.
- **weights** (*str, optional*) –Path to the checkpoint. If it is not specified and model is a model name of metafile, the weights will be loaded from metafile. Defaults to None.
- **device** (*str, optional*) –Device to run inference. If None, the available device will be automatically used. Defaults to None.
- **scope** (*str, optional*) –The scope of the model. Defaults to “mmocr”.

返回类型 None

pred2dict(*data_sample*)

Extract elements necessary to represent a prediction into a dictionary. It’s better to contain only basic data elements such as strings and numbers in order to guarantee it’s json-serializable.

参数 **data_sample** (TextDetDataSample) –The data sample to be converted.

返回 The output dictionary.

返回类型 dict

42.1.3 TextRecInferencer

class mmocr.apis.inferencers.**TextRecInferencer**(*model=None*, *weights=None*, *device=None*, *scope='mmocr'*)

Text Recognition inferencer.

参数

- **model** (*str, optional*) –Path to the config file or the model name defined in metafile. For example, it could be "crnn_mini-vgg_5e_mj" or "configs/textrecog/crnn/crnn_mini-vgg_5e_mj.py". If model is not specified, user must provide the *weights* saved by MMEngine which contains the config string. Defaults to None.
- **weights** (*str, optional*) –Path to the checkpoint. If it is not specified and model is a model name of metafile, the weights will be loaded from metafile. Defaults to None.
- **device** (*str, optional*) –Device to run inference. If None, the available device will be automatically used. Defaults to None.
- **scope** (*str, optional*) –The scope of the model. Defaults to "mmocr".

返回类型 None

pred2dict(*data_sample*)

Extract elements necessary to represent a prediction into a dictionary. It's better to contain only basic data elements such as strings and numbers in order to guarantee it's json-serializable.

参数 **data_sample** (TextRecogDataSample) –The data sample to be converted.

返回 The output dictionary.

返回类型 dict

42.1.4 TextSpotInferencer

class mmocr.apis.inferencers.**TextSpotInferencer**(*model=None*, *weights=None*, *device=None*, *scope='mmocr'*)

Text Spotting inferencer.

参数

- **model** (*str, optional*) –Path to the config file or the model name defined in metafile. For example, it could be "dbnet_resnet18_fpnc_1200e_icdar2015" or "configs/textdet/dbnet/dbnet_resnet18_fpnc_1200e_icdar2015.py". If model is not specified, user must provide the *weights* saved by MMEngine which contains the config string. Defaults to None.
- **weights** (*str, optional*) –Path to the checkpoint. If it is not specified and model is a model name of metafile, the weights will be loaded from metafile. Defaults to None.

- **device** (*str, optional*) –Device to run inference. If None, the available device will be automatically used. Defaults to None.
- **scope** (*str, optional*) –The scope of the model. Defaults to "mmocr".

返回类型 None

pred2dict(*data_sample*)

Extract elements necessary to represent a prediction into a dictionary. It's better to contain only basic data elements such as strings and numbers in order to guarantee it's json-serializable.

参数 **data_sample** (*TextSpottingDataSample*) –The data sample to be converted.

返回 The output dictionary.

返回类型 dict

42.1.5 KIEInferencer

class mmocr.apis.inferencers.**KIEInferencer**(*model=None*, *weights=None*, *device=None*, *scope='mmocr'*)

Key Information Extraction Inferencer.

参数

- **model** (*str, optional*) –Path to the config file or the model name defined in metafile. For example, it could be "sdmgr_unet16_60e_wildreceipt" or "configs/kie/sdmgr/sdmgr_unet16_60e_wildreceipt.py". If model is not specified, user must provide the *weights* saved by MMEngine which contains the config string. Defaults to None.
- **weights** (*str, optional*) –Path to the checkpoint. If it is not specified and model is a model name of metafile, the weights will be loaded from metafile. Defaults to None.
- **device** (*str, optional*) –Device to run inference. If None, the available device will be automatically used. Defaults to None.
- **scope** (*str, optional*) –The scope of the model. Defaults to "mmocr".

返回类型 None

static kie_collate(*data_batch*)

A collate function designed for KIE, where the first element (input) is a dict and we only want to keep it as-is instead of batching elements inside.

返回 Transversed Data in the same format as the data_itement of `data_batch`.

返回类型 Any

参数 **data_batch** (*Sequence*) –

pred2dict(*data_sample*)

Extract elements necessary to represent a prediction into a dictionary. It's better to contain only basic data elements such as strings and numbers in order to guarantee it's json-serializable.

参数 **data_sample** (TextRecogDataSample) –The data sample to be converted.

返回 The output dictionary.

返回类型 dict

visualize(*inputs*, *preds*, *return_vis=False*, *show=False*, *wait_time=0*, *draw_pred=True*, *pred_score_thr=0.3*, *save_vis=False*, *img_out_dir=''*)

Visualize predictions.

参数

- **inputs** (*List[Union[str, np.ndarray]]*) –Inputs for the inferencer.
- **preds** (*List[Dict]*) –Predictions of the model.
- **return_vis** (*bool*) –Whether to return the visualization result. Defaults to False.
- **show** (*bool*) –Whether to display the image in a popup window. Defaults to False.
- **wait_time** (*float*) –The interval of show (s). Defaults to 0.
- **draw_pred** (*bool*) –Whether to draw predicted bounding boxes. Defaults to True.
- **pred_score_thr** (*float*) –Minimum score of bboxes to draw. Defaults to 0.3.
- **save_vis** (*bool*) –Whether to save the visualization result. Defaults to False.
- **img_out_dir** (*str*) –Output directory of visualization results. If left as empty, no file will be saved. Defaults to ''.

返回 Returns visualization results only if applicable.

返回类型 List[np.ndarray] or None

CHAPTER 43

mmocr.structures

TextDetDataSample	A data structure interface of MMOCR.
TextRecogDataSample	A data structure interface of MMOCR for text recognition.
KIEDataSample	A data structure interface of MMOCR.

43.1 TextDetDataSample

class mmocr.structures.**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))
>>> 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),
```

(下页继续)

(续上页)

```
...                         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

43.2 TextRecogDataSample

class mmocr.structures.**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)
```

(下页继续)

(续上页)

```
>>> 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)
>>> 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

43.3 KIEDataSample

class mmocr.structures.**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),
...                 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]])
```

(下页继续)

(续上页)

```
        ) 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 44

mmocr.datasets

mmocr.datasets

44.1 Samplers

`BatchAugSampler`	Sampler that repeats the same data elements for num_repeats times.

44.1.1 BatchAugSampler

class mmocr.datasets.samplers.**BatchAugSampler**(*dataset*, *shuffle=True*, *num_repeats=3*, *seed=None*)

Sampler that repeats the same data elements for num_repeats times. The batch size should be divisible by num_repeats.

It ensures that different each augmented version of a sample will be visible to a different process (GPU). Heavily based on torch.utils.data.DistributedSampler.

This sampler was modified from https://github.com/facebookresearch/deit/blob/0c4b8f60/samplers.py Used in Copyright (c) 2015-present, Facebook, Inc.

参数

- **dataset** (*Sized*) –The dataset.
- **shuffle** (*bool*) –Whether shuffle the dataset or not. Defaults to True.
- **num_repeats** (*int*) –The repeat times of every sample. Defaults to 3.
- **seed** (*int, optional*) –Random seed used to shuffle the sampler if `shuffle=True`. This number should be identical across all processes in the distributed group. Defaults to None.

set_epoch(*epoch*)

Sets the epoch for this sampler.

When `shuffle=True`, this ensures all replicas use a different random ordering for each epoch. Otherwise, the next iteration of this sampler will yield the same ordering.

参数 **epoch** (*int*) –Epoch number.

返回类型 None

44.2 Datasets

OCRDataset	OCRDataset for text detection and text recognition.
WildReceiptDataset	WildReceipt Dataset for key information extraction.

44.2.1 OCRDataset

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*) –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. `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}

44.2.2 WildReceiptDataset

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,
```

(下页继续)

(续上页)

```
                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

44.3 Compatible Datasets

IcdarDataset	Dataset for text detection while ann_file in coco format.
RecogLMDBDataset	RecogLMDBDataset for text recognition.
RecogTextDataset	RecogTextDataset for text recognition.

44.3.1 IcdarDataset

class mmocr.datasets.**IcdarDataset**(**args*, *seg_map_suffix='.png'*, *proposal_file=None*, *file_client_args=None*, *backend_args=None*, *return_classes=False*, ***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]]

44.3.2 RecogLMDBDataset

class mmocr.datasets.**RecogLMDBDataset**(*ann_file=''*, *img_color_type='color'*, *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. The lmdb file should contain three keys: 'num-samples', 'label-xxxxxxxxx' and 'image-xxxxxxxxx', where 'xxxxxxxxx' is the index of the image. The value of 'num-samples' is the total number of images. The value of 'label-xxxxxxx' is the text label of the image, and the value of 'image-xxxxxxx' is the image data.

following keys: Each item fetched from this dataset will be a dict containing the following keys:

- img (ndarray): The loaded image.

- img_path (str): The image key.
- instances (list[dict]): The list of annotations for the image.

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ‘’.
- **img_color_type** (*str*) –The flag argument for :func:`mmcv.imfrombytes`, which determines how the image bytes will be parsed. Defaults to ‘color’.
- **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)

prepare_data(*idx*)

Get data processed by `self.pipeline`.

参数 **idx** (*int*) –The index of `data_info`.

返回 Depends on `self.pipeline`.

返回类型 Any

44.3.3 RecogTextDataset

class mmocr.datasets.**RecogTextDataset**(*ann_file=''*, *backend_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 `''`.
- **backend_args** (*dict, optional*) –Arguments to instantiate the prefix of uri corresponding backend. Defaults to 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)

44.4 Dataset Wrapper

ConcatDataset	A wrapper of concatenated dataset.

44.4.1 ConcatDataset

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.

CHAPTER 45

mmocr.datasets

mmocr.datasets.transforms

45.1 Loading

`LoadImageFromFile`	Load an image from file.
`LoadOCRAnnotations`	Load and process the `instances` annotation provided by dataset.
`LoadKIEAnnotations`	Load and process the `instances` annotation provided by dataset.

45.1.1 LoadImageFromFile

class mmocr.datasets.transforms.**LoadImageFromFile**(*to_float32=False*, *color_type='color'*, *imdecode_backend='cv2'*, *file_client_args=None*, *min_size=0*, *ignore_empty=False*, ***, *backend_args=None*)

Load an image from file.

Required Keys:

- img_path

Modified Keys:

- img
- img_shape
- ori_shape

参数

- **to_float32** (*bool*) –Whether to convert the loaded image to a float32 numpy array. If set to False, the loaded image is an uint8 array. Defaults to False.
- **color_type** (*str*) –The flag argument for :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 None. It will be deprecated in future. Please use backend_args instead. Deprecated in version 1.0.0rc6.
- **backend_args** (*dict, optional*) –Instantiates the corresponding file backend. It may contain *backend* key to specify the file backend. If it contains, the file backend corresponding to this value will be used and initialized with the remaining values, otherwise the corresponding file backend will be selected based on the prefix of the file path. Defaults to None. New in version 1.0.0rc6.
- **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 The dict contains loaded image and meta information.

返回类型 dict

45.1.2 LoadOCRAnnotations

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

45.1.3 LoadKIEAnnotations

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),
    # In np.int64 type.
    'gt_bboxes_labels': np.ndarray(N, ),
    # In np.int32 type.
```

(下页继续)

(续上页)

```
    '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

45.2 TextDet Transforms

BoundedScaleAspectJitter	First randomly rescale the image so that the longside and shortside of the image are around the bound; then jitter its aspect ratio.
RandomFlip	Flip the image & bbox polygon.
SourceImagePad	Pad Image to target size.
ShortScaleAspectJitter	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.
TextDetRandomCrop	Randomly select a region and crop images to a target size and make sure to contain text region.
TextDetRandomCropFlip	Random crop and flip a patch in the image.

45.2.1 BoundedScaleAspectJitter

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

45.2.2 RandomFlip

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

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'.

- **swap_seg_labels** (*Optional[Sequence]*) –

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]

45.2.3 SourceImagePad

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)

45.2.4 ShortScaleAspectJitter

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*) –

45.2.5 TextDetRandomCrop

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*) –

45.2.6 TextDetRandomCropFlip

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

45.3 TextRecog Transforms

TextRecogGeneralAug	A general geometric augmentation tool for text images in the CVPR 2020 paper "Learn to Augment: Joint Data Augmentation and Network Optimization for Text Recognition".
CropHeight	Randomly crop the image's height, either from top or bottom.
ImageContentJitter	Jitter the image contents.
ReversePixels	Reverse image pixels.
PyramidRescale	Resize the image to the base shape, downsample it with gaussian pyramid, and rescale it back to original size.
PadToWidth	Only pad the image's width.
RescaleToHeight	Rescale the image to the height according to setting and keep the aspect ratio unchanged if possible.

45.3.1 TextRecogGeneralAug

class mmocr.datasets.transforms.**TextRecogGeneralAug**

A general geometric augmentation tool for text images in the CVPR 2020 paper "Learn to Augment: Joint Data Augmentation and Network Optimization for Text Recognition". It applies distortion, stretching, and perspective transforms to an image.

This implementation is adapted from https://github.com/RubanSeven/Text-Image-Augmentation-python/blob/master/augment.py # noqa

TODO: Split this transform into three transforms.

Required Keys:

- img

Modified Keys:

- img
- img_shape

tia_distort(*img*, *segment=4*)

Image distortion.

参数

- **img** (*np.ndarray*) –The image.
- **segment** (*int*) –The number of segments to divide the image along the width. Defaults to 4.

返回类型 numpy.ndarray

tia_perspective(*img*)

Image perspective transformation.

参数

- **img** (*np.ndarray*) –The image.
- **segment** (*int*) –The number of segments to divide the image along the width. Defaults to 4.

返回类型 numpy.ndarray

tia_stretch(*img*, *segment=4*)

Image stretching.

参数

- **img** (*np.ndarray*) –The image.

- **segment** (*int*) –The number of segments to divide the image along the width. Defaults to 4.

返回类型 numpy.ndarray

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

warp_mls(*src*, *src_pts*, *dst_pts*, *dst_w*, *dst_h*, *trans_ratio=1.0*)

Warp the image.

参数

- **src** (*numpy.ndarray*) –
- **src_pts** (*List[int]*) –
- **dst_pts** (*List[int]*) –
- **dst_w** (*int*) –
- **dst_h** (*int*) –
- **trans_ratio** (*float*) –

返回类型 numpy.ndarray

45.3.2 CropHeight

class mmocr.datasets.transforms.**CropHeight**(*min_pixels=1*, *max_pixels=8*)

Randomly crop the image's height, either from top or bottom.

Adapted from https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.6/ppocr/data/imaug/rec_img_aug.py # noqa

Required Keys:

- img

Modified Keys:

- img
- img_shape

参数

- **crop_min** (*int*) –Minimum pixel(s) to crop. Defaults to 1.

- **crop_max** (*int*) –Maximum pixel(s) to crop. Defaults to 8.
- **min_pixels** (*int*) –
- **max_pixels** (*int*) –

返回类型 None

transform(*results*)

Transform function to crop images.

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

返回 Cropped results.

返回类型 dict

45.3.3 ImageContentJitter

class mmocr.datasets.transforms.**ImageContentJitter**

Jitter the image contents.

Adapted from https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.6/ppocr/data/imaug/rec_img_aug.py # noqa

Required Keys:

- img

Modified Keys:

- img

transform(*results*, *jitter_ratio=0.01*)

Transform function to jitter images.

参数

- **results** (*dict*) –Result dict from loading pipeline.
- **jitter_ratio** (*float*) –Controls the strength of jittering. Defaults to 0.01.

返回 Jittered results.

返回类型 dict

45.3.4 ReversePixels

class mmocr.datasets.transforms.**ReversePixels**

Reverse image pixels.

Adapted from https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.6/ppocr/data/imaug/rec_img_aug.py # noqa

Required Keys:

- img

Modified Keys:

- img

transform(*results*)

Transform function to reverse image pixels.

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

返回 Reversed results.

返回类型 dict

45.3.5 PyramidRescale

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

45.3.6 PadToWidth

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

45.3.7 RescaleToHeight

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

45.4 OCR Transforms

RandomCrop	Randomly crop images and make sure to contain at least one intact instance.
RandomRotate	Randomly rotate the image, boxes, and polygons.
Resize	Resize image & bboxes & polygons.
FixInvalidPolygon	Fix invalid polygons in the dataset.
RemoveIgnored	Removed ignored elements from the pipeline.

45.4.1 RandomCrop

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*) –

45.4.2 RandomRotate

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

45.4.3 Resize

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

45.4.4 FixInvalidPolygon

class mmocr.datasets.transforms.**FixInvalidPolygon**(*mode='fix'*, *min_poly_points=4*, *fix_from_bbox=True*)

Fix invalid polygons in the dataset.

Required Keys:

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

Modified Keys:

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

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 or converting the bboxes to polygons. If it can’t be fixed by any means (e.g. the polygon contains less than 3 points or it’s actually a line/point), the annotation will be removed. For the ‘ignore’ mode, the invalid polygons will be set to “ignored” during training. Defaults to ‘fix’.
- **min_poly_points** (*int*) –Minimum number of the coordinate points in a polygon. Defaults to 4.
- **fix_from_bbox** (*bool*) –Whether to convert the bboxes to polygons when the polygon is invalid and not directly fixable. Defaults to True.

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. If all the polygons are unfixable, return None.

返回类型 Optional[dict]

45.4.5 RemoveIgnored

class mmocr.datasets.transforms.**RemoveIgnored**

Removed ignored elements from the pipeline.

Required Keys:

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

- gt_texts (optional)

Modified Keys:

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

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

45.5 Formatting

PackTextDetInputs	Pack the inputs data for text detection.
PackTextRecogInputs	Pack the inputs data for text recognition.
PackKIEInputs	Pack the inputs data for key information extraction.

45.5.1 PackTextDetInputs

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

45.5.2 PackTextRecogInputs

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

45.5.3 PackKIEInputs

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

45.6 Transform Wrapper

ImgAugWrapper	A wrapper around imgaug https://github.com/aleju/imgaug.
TorchVisionWrapper	A wrapper around torchvision transforms.

45.6.1 ImgAugWrapper

class mmocr.datasets.transforms.**ImgAugWrapper**(*args=None*, *fix_poly_trans={'type': 'FixInvalidPolygon'}*)

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.

- **fix_poly_trans** (*dict*) –The transform configuration to fix invalid polygons. Set it to None if no fixing is needed. Defaults to dict(type='FixInvalidPolygon').

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

45.6.2 TorchVisionWrapper

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

A wrapper around torchvision transforms. 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

45.7 Adapter

MMDet2MMOCR	Convert transforms’ s data format from MMDet to MMOCR.
MMOCR2MMDet	Convert transforms’ s data format from MMOCR to MMDet.

45.7.1 MMDet2MMOCR

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)

45.7.2 MMOCR2MMDet

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)

CHAPTER 46

mmocr.models

46.1 models.common

46.1.1 BackBones

UNet	UNet backbone.

UNet

class mmocr.models.common.**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.

46.1.2 Dictionary

Dictionary	The class generates a dictionary for recognition.

Dictionary

class mmocr.models.common.**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

46.1.3 Losses

MaskedBalancedBCEWithLogitsLoss	This loss combines a Sigmoid layers and a masked balanced BCE loss in one single class.
MaskedDiceLoss	Masked dice loss.
MaskedSmoothL1Loss	Masked Smooth L1 loss.
MaskedSquareDiceLoss	Masked square dice loss.
MaskedBCEWithLogitsLoss	This loss combines a Sigmoid layers and a masked BCE loss in one single class.
SmoothL1Loss	Smooth L1 loss.
CrossEntropyLoss	Cross entropy loss.
MaskedBalancedBCELoss	Masked Balanced BCE loss.
MaskedBCELoss	Masked BCE loss.

MaskedBalancedBCEWithLogitsLoss

class mmocr.models.common.**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

MaskedDiceLoss

class mmocr.models.common.**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

MaskedSmoothL1Loss

class mmocr.models.common.**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

MaskedSquareDiceLoss

class mmocr.models.common.**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

MaskedBCEWithLogitsLoss

class mmocr.models.common.**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

SmoothL1Loss

class mmocr.models.common.**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

CrossEntropyLoss

class mmocr.models.common.**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

MaskedBalancedBCELoss

class mmocr.models.common.**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

MaskedBCELoss

class mmocr.models.common.**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

46.1.4 Layers

TFEncoderLayer	Transformer Encoder Layer.
TFDecoderLayer	Transformer Decoder Layer.

TFEncoderLayer

class mmocr.models.common.**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.

TFDecoderLayer

class mmocr.models.common.**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.

46.1.5 Modules

ScaledDotProductAttention	Scaled Dot-Product Attention Module.
MultiHeadAttention	Multi-Head Attention module.
PositionwiseFeedForward	Two-layer feed-forward module.
PositionalEncoding	Fixed positional encoding with sine and cosine functions.

ScaledDotProductAttention

class mmocr.models.common.**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.

MultiHeadAttention

class mmocr.models.common.**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.

PositionwiseFeedForward

class mmocr.models.common.**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.

PositionalEncoding

class mmocr.models.common.**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, ···)

46.2 models.textdet

46.2.1 Detectors

SingleStageTextDetector	The class for implementing single stage text detector.
DBNet	The class for implementing DBNet text detector: Real-time Scene Text Detection with Differentiable Binarization.
PANet	The class for implementing PANet text detector:
PSENet	The class for implementing PSENet text detector: Shape Robust Text Detection with Progressive Scale Expansion Network.
TextSnake	The class for implementing TextSnake text detector: TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.
FCENet	The class for implementing FCENet text detector FCENet(CVPR2021): Fourier Contour Embedding for Arbitrary-shaped Text Detection
DRRG	The class for implementing DRRG text detector.
MMDetWrapper	A wrapper of MMDet’s model.

SingleStageTextDetector

class mmocr.models.textdet.**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*]

DBNet

class mmocr.models.textdet.**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

PANet

class mmocr.models.textdet.**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

PSENet

class mmocr.models.textdet.**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

TextSnake

class mmocr.models.textdet.**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

FCENet

class mmocr.models.textdet.**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

DRRG

class mmocr.models.textdet.**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

MMDetWrapper

class mmocr.models.textdet.**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]

46.2.2 Data Preprocessors

TextDetDataPreprocessor	Image pre-processor for detection tasks.

TextDetDataPreprocessor

class mmocr.models.textdet.**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

46.2.3 Necks

FPEM_FFM	This code is from https://github.com/WenmuZhou/PAN.pytorch.
FPNF	FPN-like fusion module in Shape Robust Text Detection with Progressive Scale Expansion Network.
FPNC	FPN-like fusion module in Real-time Scene Text Detection with Differentiable Binarization.
FPN_UNet	The class for implementing DRRG and TextSnake U-Net-like FPN.

FPEM_FFM

class mmocr.models.textdet.**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.

返回类型 None

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 tuple[Tensor]

FPNF

class mmocr.models.textdet.**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

FPNC

class mmocr.models.textdet.**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

FPN_UNet

class mmocr.models.textdet.**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

46.2.4 Heads

BaseTextDetHead	Base head for text detection, build the loss and postprocessor.
PSEHead	The class for PSENet head.
PANHead	The class for PANet head.
DBHead	The class for DBNet head.
FCEHead	The class for implementing FCENet head.
TextSnakeHead	The class for TextSnake head: TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.
DRRGHead	The class for DRRG head: Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection.

BaseTextDetHead

class mmocr.models.textdet.**BaseTextDetHead**(*module_loss=None*, *postprocessor=None*, *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, 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.
- **module_loss** (*Optional[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

PSEHead

class mmocr.models.textdet.**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

PANHead

class mmocr.models.textdet.**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

DBHead

class mmocr.models.textdet.**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

FCEHead

class mmocr.models.textdet.**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

TextSnakeHead

class mmocr.models.textdet.**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

DRRGHead

class mmocr.models.textdet.**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)

46.2.5 Module Losses

SegBasedModuleLoss	Base class for the module loss of segmentation-based text detection algorithms with some handy utilities.
PANModuleLoss	The class for implementing PANet loss.
PSEModuleLoss	The class for implementing PSENet loss.
DBModuleLoss	The class for implementing DBNet loss.
TextSnakeModuleLoss	The class for implementing TextSnake loss.
FCEModuleLoss	The class for implementing FCENet loss.
DRRGModuleLoss	The class for implementing DRRG loss.

SegBasedModuleLoss

class mmocr.models.textdet.**SegBasedModuleLoss**

Base class for the module loss of segmentation-based text detection algorithms with some handy utilities.

返回类型 None

PANModuleLoss

class mmocr.models.textdet.**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)

PSEModuleLoss

class mmocr.models.textdet.**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

DBModuleLoss

class mmocr.models.textdet.**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

TextSnakeModuleLoss

class mmocr.models.textdet.**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

FCEModuleLoss

class mmocr.models.textdet.**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]

DRRGModuleLoss

class mmocr.models.textdet.**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': 'MaskedBalancedBCEWithLogitsLoss'}*, *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

46.2.6 Postprocessors

BaseTextDetPostProcessor	Base postprocessor for text detection models.
PSEPostprocessor	Decoding predictions of PSENet to instances.
PANPostprocessor	Convert scores to quadrangles via post processing in PANet.
DBPostprocessor	Decoding predictions of DbNet to instances.
DRRGPostprocessor	Merge text components and construct boundaries of text instances.
FCEPostprocessor	Decoding predictions of FCENet to instances.
TextSnakePostprocessor	Decoding predictions of TextSnake to instances.

BaseTextDetPostProcessor

class mmocr.models.textdet.**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]]

PSEPostprocessor

class mmocr.models.textdet.**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*

PANPostprocessor

class mmocr.models.textdet.**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*

DBPostprocessor

class mmocr.models.textdet.**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*

DRRGPostprocessor

class mmocr.models.textdet.**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]

FCEPostprocessor

class mmocr.models.textdet.**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]]

TextSnakePostprocessor

class mmocr.models.textdet.**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]

46.3 models.textrecog

46.3.1 Recognizers

BaseRecognizer	Base class for recognizer.
EncoderDecoderRecognizer	Base class for encode-decode recognizer.
CRNN	CTC-loss based recognizer.
SARNet	Implementation of SAR
NRTR	Implementation of NRTR
RobustScanner	Implementation of ‘RobustScanner.
SATRN	Implementation of SATRN
ABINet	Implementation of ‘Read Like Humans: Autonomous, Bidirectional and Iterative LanguageModeling for Scene Text Recognition.
MASTER	Implementation of MASTER
ASTER	Implement ‘ASTER: An Attentional Scene Text Recognizer with Flexible Rectification.

BaseRecognizer

class mmocr.models.textrecog.**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

EncoderDecoderRecognizer

class mmocr.models.textrecog.**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*]

CRNN

class mmocr.models.textrecog.**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

SARNet

class mmocr.models.textrecog.**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

NRTR

class mmocr.models.textrecog.**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

RobustScanner

class mmocr.models.textrecog.**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

SATRN

class mmocr.models.textrecog.**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

ABINet

class mmocr.models.textrecog.**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

MASTER

class mmocr.models.textrecog.**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

ASTER

class mmocr.models.textrecog.**ASTER**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *data_preprocessor=None*, *init_cfg=None*)

Implement ‘ASTER: An Attentional Scene Text Recognizer with Flexible Rectification.

<https://ieeexplore.ieee.org/abstract/document/8395027/‘

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

46.3.2 Data Preprocessors

TextRecogDataPreprocessor	Image pre-processor for recognition tasks.

TextRecogDataPreprocessor

class mmocr.models.textrecog.**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

46.3.3 Preprocessors

STN	Implement STN module in ASTER: An Attentional Scene Text Recognizer with Flexible Rectification (https://ieeexplore.ieee.org/abstract/document/8395027/)

STN

class mmocr.models.textrecog.**STN**(*in_channels*, *resized_image_size=(32, 64)*, *output_image_size=(32, 100)*, *num_control_points=20*, *margins=[0.05, 0.05]*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}, {'type': 'Constant', 'val': 1, 'layer': 'BatchNorm2d'}]*)

Implement STN module in ASTER: An Attentional Scene Text Recognizer with Flexible Rectification (https://ieeexplore.ieee.org/abstract/document/8395027/)

参数

- **in_channels** (*int*) –The number of input channels.
- **resized_image_size** (*Tuple[int, int]*) –The resized image size. The input image will be downsampled to have a better recitified result.
- **output_image_size** (*Tuple[int, int]*) –The size of the output image for TPS. Defaults to (32, 100).
- **num_control_points** (*int*) –The number of control points. Defaults to 20.
- **margins** (*Tuple[float, float]*) –The margins for control points to the top and down side of the image for TPS. Defaults to [0.05, 0.05].

- **init_cfg** (*Optional[Union[Dict, List[Dict]]]*) –

forward(*img*)

Forward function of STN.

参数 **img** (*Tensor*) –The input image tensor.

返回 The rectified image tensor.

返回类型 Tensor

init_stn(*stn_fc2*)

Initialize the output linear layer of stn, so that the initial source point will be at the top and down side of the image, which will help to optimize.

参数 **stn_fc2** (*nn.Linear*) –The output linear layer of stn.

返回类型 None

46.3.4 BackBones

ResNet31OCR	Implement ResNet backbone for text recognition, modified from
MiniVGG	A mini VGG backbone for text recognition, modified from ‘VGG-VeryDeep.
NRTRModalityTransform	Modality transform in NRTR.
ShallowCNN	Implement Shallow CNN block for SATRN.
ResNetABI	Implement ResNet backbone for text recognition, modified from ‘ResNet.
ResNet	**param in_channels** Number of channels of input image tensor.
MobileNetV2	See mmdet.models.backbones.MobileNetV2 for details.

ResNet31OCR

class mmocr.models.textrecog.**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.

MiniVGG

class mmocr.models.textrecog.**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

NRTRModalityTransform

class mmocr.models.textrecog.**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

ShallowCNN

class mmocr.models.textrecog.**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

ResNetABI

class mmocr.models.textrecog.**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]

ResNet

class mmocr.models.textrecog.**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

MobileNetV2

class mmocr.models.textrecog.**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

46.3.5 Encoders

SAREncoder	Implementation of encoder module in 'SAR.
NRTREncoder	Transformer Encoder block with self attention mechanism.
BaseEncoder	Base Encoder class for text recognition.
ChannelReductionEncoder	Change the channel number with a one by one convolutional layer.
SATRNEncoder	Implement encoder for SATRN, see 'SATRN.
ABIEncoder	Implement transformer encoder for text recognition, modified from *<https://github.com/FangShancheng/ABINet>*.
ASTEREncoder	Implement BiLSTM encoder module in 'ASTER: An Attentional Scene Text Recognizer with Flexible Rectification.

SAREncoder

class mmocr.models.textrecog.**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

NRTREncoder

class mmocr.models.textrecog.**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

BaseEncoder

class mmocr.models.textrecog.**BaseEncoder**(*init_cfg=None*)

Base Encoder class for text recognition.

参数 **init_cfg**(*Optional[Union[dict, List[dict]]]*) –

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.

ChannelReductionEncoder

class mmocr.models.textrecog.**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

SATRNEncoder

class mmocr.models.textrecog.**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

ABIEncoder

class mmocr.models.textrecog.**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

ASTEREncoder

class mmocr.models.textrecog.**ASTEREncoder**(*in_channels*, *num_layers=2*, *init_cfg={'layer': 'Conv2d', 'type': 'Xavier'}*)

Implement BiLSTM encoder module in ‘ASTER: An Attentional Scene Text Recognizer with Flexible Rectification.

<https://ieeexplore.ieee.org/abstract/document/8395027/‘ :param in_channels: Number of input channels. :type in_channels: int :param num_layers: Layers of BiLSTM. Defaults to 2. :type num_layers: int

参数

- **in_channels** (*int*) –
- **num_layers** (*int*) –

返回类型 None

forward(*feat*, *img_metas=None*)

参数 **feat** (*Tensor*) –Feature of shape (N, C, 1, W).

返回 Output of BiLSTM.

返回类型 Tensor

46.3.6 Decoders

BaseDecoder	Base decoder for text recognition, build the loss and post-processor.
ABILanguageDecoder	Transformer-based language model responsible for spell correction. Implementation of language model of ABINet.
ABIVisionDecoder	Converts visual features into text characters.
ABIFuser	A special decoder responsible for mixing and aligning visual feature and linguistic feature.
CRNNDecoder	Decoder for CRNN.
ParallelSARDecoder	Implementation Parallel Decoder module in ‘SAR.
SequentialSARDecoder	Implementation Sequential Decoder module in ‘SAR.
ParallelSARDecoderWithBS	Parallel Decoder module with beam-search in SAR.
NRTRDecoder	Transformer Decoder block with self attention mechanism.
SequenceAttentionDecoder	Sequence attention decoder for RobustScanner.
PositionAttentionDecoder	Position attention decoder for RobustScanner.
RobustScannerFuser	Decoder for RobustScanner.
MasterDecoder	Decoder module in MASTER.
ASTERDecoder	Implement attention decoder.

BaseDecoder

class mmocr.models.textrecog.**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*]

ABILanguageDecoder

class mmocr.models.textrecog.**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

ABIVisionDecoder

class mmocr.models.textrecog.**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

ABIFuser

class mmocr.models.textrecog.**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

CRNNDecoder

class mmocr.models.textrecog.**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

ParallelSARDecoder

class mmocr.models.textrecog.**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 $(N, self.max_seq_len, C)$ 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

SequentialSARDecoder

class mmocr.models.textrecog.**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

ParallelSARDecoderWithBS

class mmocr.models.textrecog.**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

NRTRDecoder

class mmocr.models.textrecog.**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

SequenceAttentionDecoder

class mmocr.models.textrecog.**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

PositionAttentionDecoder

class mmocr.models.textrecog.**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

RobustScannerFuser

class mmocr.models.textrecog.**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

MasterDecoder

class mmocr.models.textrecog.**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

ASTERDecoder

class mmocr.models.textrecog.**ASTERDecoder**(*in_channels*, *emb_dims=512*, *attn_dims=512*, *hidden_size=512*, *dictionary=None*, *max_seq_len=25*, *module_loss=None*, *postprocessor=None*, *init_cfg={'layer': 'Conv2d', 'type': 'Xavier'}*)

Implement attention decoder.

参数

- **in_channels** (*int*) –Number of input channels.
- **emb_dims** (*int*) –Dims of char embedding. Defaults to 512.
- **attn_dims** (*int*) –Dims of attention. Both hidden states and features will be projected to this dims. Defaults to 512.
- **hidden_size** (*int*) –Dims of hidden state for GRU. Defaults to 512.
- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*. Defaults to None.
- **max_seq_len** (*int*) –Maximum output sequence length T. Defaults to 25.
- **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.

forward_test(*feat=None*, *out_enc=None*, *data_samples=None*)

参数

- **feat** (*Tensor*) –Feature from backbone. Unused in this decoder.
- **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. Unused in this decoder.

返回 The raw logit tensor. Shape (N, T, C) where C is `num_classes`.

返回类型 Tensor

forward_train(*feat=None*, *out_enc=None*, *data_samples=None*)

参数

- **feat** (*Tensor*) –Feature from backbone. Unused in this decoder.
- **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, T, C) where C is `num_classes`.

返回类型 Tensor

46.3.7 Module Losses

BaseTextRecogModuleLoss	Base recognition loss.
CEModuleLoss	Implementation of loss module for encoder-decoder based text recognition method with CrossEntropy loss.
CTCModuleLoss	Implementation of loss module for CTC-loss based text recognition.
ABIModuleLoss	Implementation of ABINet multiloss that allows mixing different types of losses with weights.

BaseTextRecogModuleLoss

class mmocr.models.textrecog.**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*]

CEModuleLoss

class mmocr.models.textrecog.**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

CTCModuleLoss

class mmocr.models.textrecog.**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*]

ABIModuleLoss

class mmocr.models.textrecog.**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

46.3.8 Postprocessors

BaseTextRecogPostprocessor	Base text recognition postprocessor.
AttentionPostprocessor	PostProcessor for seq2seq.
CTCPostProcessor	PostProcessor for CTC.

BaseTextRecogPostprocessor

class mmocr.models.textrecog.**BaseTextRecogPostprocessor**(*dictionary*, *max_seq_len=40*, *ignore_chars=['padding']*, ***kwargs*)

Base text recognition postprocessor.

参数

- **dictionary** (dict or `Dictionary`) –The config for *Dictionary* or the instance of *Dictionary*.
- **max_seq_len** (*int*) –max_seq_len (int): Maximum sequence length. The sequence is usually generated from decoder. Defaults to 40.
- **ignore_chars** (*list[str]*) –A list of characters to be ignored from the final results. Postprocessor will skip over these characters when converting raw indexes to characters. Apart from single characters, each item can be one of the following reversed keywords: 'padding', 'end' and 'unknown', which refer to their corresponding special tokens in the dictionary.

返回类型 None

get_single_prediction(*probs*, *data_sample=None*)

Convert the output probabilities of a single image to index and score.

参数

- **probs** (*torch.Tensor*) –Character probabilities with shape (T, C).

- **data_sample** (TextRecogDataSample) –Datasample of an image.

返回 Index and scores per-character.

返回类型 tuple(list[int], list[float])

AttentionPostprocessor

class mmocr.models.textrecog.**AttentionPostprocessor**(*dictionary*, *max_seq_len=40*, *ignore_chars=['padding']*, ***kwargs*)

PostProcessor for seq2seq.

参数

- **dictionary** (*Union[mmocr.models.common.dictionary.dictionary.Dictionary, Dict]*) –
- **max_seq_len** (*int*) –
- **ignore_chars** (*Sequence[str]*) –

返回类型 None

get_single_prediction(*probs*, *data_sample=None*)

Convert the output probabilities of a single image to index and score.

参数

- **probs** (*torch.Tensor*) –Character probabilities with shape (T, C).
- **data_sample** (TextRecogDataSample, *optional*) –Datasample of an image. Defaults to None.

返回 index and score.

返回类型 tuple(list[int], list[float])

CTCPostProcessor

class mmocr.models.textrecog.**CTCPostProcessor**(*dictionary*, *max_seq_len=40*, *ignore_chars=['padding']*, ***kwargs*)

PostProcessor for CTC.

参数

- **dictionary** (*Union[mmocr.models.common.dictionary.dictionary.Dictionary, Dict]*) –
- **max_seq_len** (*int*) –
- **ignore_chars** (*Sequence[str]*) –

返回类型 None

get_single_prediction(*probs*, *data_sample*)

Convert the output probabilities of a single image to index and score.

参数

- **probs** (*torch.Tensor*) –Character probabilities with shape (T, C).
- **data_sample** (TextRecogDataSample) –Datasample of an image.

返回 index and score.

返回类型 tuple(list[int], list[float])

46.3.9 Layers

BidirectionalLSTM	
Adaptive2DPositionalEncoding	Implement Adaptive 2D positional encoder for SATRN, see ‘SATRN.
BasicBlock	
Bottleneck	
RobustScannerFusionLayer	
DotProductAttentionLayer	
PositionAwareLayer	
SATRNEncoderLayer	Implement encoder layer for SATRN, see ‘SATRN.

BidirectionalLSTM

class mmocr.models.textrecog.**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.

Adaptive2DPositionalEncoding

class mmocr.models.textrecog.**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

BasicBlock

class mmocr.models.textrecog.**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]

Bottleneck

class mmocr.models.textrecog.**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.

RobustScannerFusionLayer

class mmocr.models.textrecog.**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.

DotProductAttentionLayer

class mmocr.models.textrecog.**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.

PositionAwareLayer

class mmocr.models.textrecog.**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.

SATRNEncoderLayer

class mmocr.models.textrecog.**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

46.4 models.kie

46.4.1 Extractors

SDMGR	The implementation of the paper: Spatial Dual-Modality Graph Reasoning for Key Information Extraction.

SDMGR

class mmocr.models.kie.**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*]

46.4.2 Heads

SDMGRHead	SDMGR Head.

SDMGRHead

class mmocr.models.kie.**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*]

46.4.3 Module Losses

SDMGRModuleLoss	The implementation the loss of key information extraction proposed in the paper: Spatial Dual-Modality Graph Reasoning for Key Information Extraction.

SDMGRModuleLoss

class mmocr.models.kie.**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)

46.4.4 Postprocessors

SDMGRPostProcessor	Postprocessor for SDMGR.

SDMGRPostProcessor

class mmocr.models.kie.**SDMGRPostProcessor**(*link_type='none'*, *key_node_idx=None*, *value_node_idx=None*)

Postprocessor for SDMGR. It converts the node and edge scores into labels and edge labels. If the link_type is not "none", it reconstructs the edge labels with different strategies specified by `link_type`, which is generally known as the "openset" mode. In "openset" mode, only the edges connecting from "key" to "value" nodes will be constructed.

参数

- **link_type** (*str*) –The type of link to be constructed. Defaults to 'none'. Options are:
  - 'none': The simplest link type involving no edge postprocessing. The edge prediction

will be returned as-is.

  - ’ one-to-one’ : One key node can be connected to one value node.
  - ’ one-to-many’ : One key node can be connected to multiple value nodes.
  - ’ many-to-one’ : Multiple key nodes can be connected to one value node.
  - ’ many-to-many’ : No restrictions on the number of edges that a key/value node can have.

- **key_node_idx** (*int, optional*) –The label index of the key node. It must be specified if `link_type` is not “none”. Defaults to None.
- **value_node_idx** (*int, optional*) –The index of the value node. It must be specified if `link_type` is not “none”. Defaults to None.

decode_edges(*node_labels*, *edge_scores*, *edge_labels*)

Reconstruct the edges and update edge scores according to `link_type`.

参数

- **data_sample** (`KIEDataSample`) –A datasample containing prediction results.
- **node_labels** (*torch.Tensor*) –
- **edge_scores** (*torch.Tensor*) –
- **edge_labels** (*torch.Tensor*) –

返回

- **edge_scores (Tensor): A float tensor of shape (N, N)** indicating the confidence scores for edge predictions.
- **edge_labels (Tensor): An integer tensor of shape (N, N)** indicating the connection between nodes. Options are 0, 1.

返回类型 tuple(Tensor, Tensor)

CHAPTER 47

mmocr.evaluation

mmocr.evaluation

47.1 Evaluator

`MultiDatasetsEvaluator`	Wrapper class to compose class: *ConcatDataset* and multiple `BaseMetric` instances.

47.1.1 MultiDatasetsEvaluator

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

47.2 TextDet Metric

HmeanIOUMetric	HmeanIOU metric.

47.2.1 HmeanIOUMetric

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

47.3 TextRecog Metric

WordMetric	Word metrics for text recognition task.
CharMetric	Character metrics for text recognition task.
OneMinusNEDMetric	One minus NED metric for text recognition task.

47.3.1 WordMetric

class mmocr.evaluation.metrics.**WordMetric**(*mode='ignore_case_symbol'*, *valid_symbol='[^A-Z^a-z^0-9^ —-🄵]'*, *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^ —-🄵]’
- **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

47.3.2 CharMetric

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

47.3.3 OneMinusNEDMetric

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^ 一-🄵]'
- **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

47.4 KIE Metric

F1Metric	Compute F1 scores.

47.4.1 F1Metric

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

CHAPTER 48

mmocr.visualization

BaseLocalVisualizer	The MMOCR Text Detection Local Visualizer.
TextDetLocalVisualizer	The MMOCR Text Detection Local Visualizer.
TextRecogLocalVisualizer	MMOCR Text Detection Local Visualizer.
TextSpottingLocalVisualizer	
KIELocalVisualizer	The MMOCR Text Detection Local Visualizer.

48.1 BaseLocalVisualizer

class mmocr.visualization.**BaseLocalVisualizer**(*name='visualizer'*, *font_families='sans-serif'*, *font_properties=None*, ***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.
- **font_families** (*Union[str, List[str]]*) –The font families of labels. Defaults to 'sans-serif'.
- **font_properties** (*Union[str, FontProperties], optional*) –The font properties of texts. The format should be a path str to font file or a *font_manager.FontProperties()* object. If you want to draw Chinese texts, you need to prepare a font file that can show Chinese characters properly. For example: *simhei.ttf*,'simsun.ttc','simkai.ttf' and so on. Then set font_properties=matplotlib.font_manager.FontProperties (fname='path/to/font_file') or font_properties='path/to/font_file' This function need mmengine version >=0.6.0. Defaults to None.

返回类型 None

get_bboxes_image(*image*, *bboxes*, *colors='g'*, *filling=False*, *line_width=0.5*, *alpha=0.5*)

Draw bboxes on image.

参数

- **image** (*np.ndarray*) –The origin image to draw. The format should be RGB.
- **bboxes** (*Union[np.ndarray, torch.Tensor]*) –The bboxes to draw.
- **colors** (*Union[str, Sequence[str]]*) –The colors of bboxes. `colors` can have the same length with bboxes or just single value. If `colors` is single value, all the bboxes will have the same colors. Refer to *matplotlib.colors* for full list of formats that are accepted. Defaults to 'g'.
- **filling** (*bool*) –Whether to fill the bboxes. Defaults to False.
- **line_width** (*Union[int, float]*) –The line width of bboxes. Defaults to 0.5.
- **alpha** (*float*) –The alpha of bboxes. Defaults to 0.5.
- **self** (*mmengine.visualization.visualizer.Visualizer*) –

返回 The image with bboxes drawn.

返回类型 np.ndarray

get_labels_image(*image*, *labels*, *bboxes*, *colors='k'*, *font_size=10*, *auto_font_size=False*, *font_families='sans-serif'*, *font_properties=None*)

Draw labels on image.

参数

- **image** (*np.ndarray*) –The origin image to draw. The format should be RGB.
- **labels** (*Union[np.ndarray, torch.Tensor]*) –The labels to draw.
- **bboxes** (*Union[np.ndarray, torch.Tensor]*) –The bboxes to draw.
- **colors** (*Union[str, Sequence[str]]*) –The colors of labels. `colors` can have the same length with labels or just single value. If `colors` is single value, all the labels will have the same colors. Refer to *matplotlib.colors* for full list of formats that are accepted. Defaults to 'k'.
- **font_size** (*Union[int, float]*) –The font size of labels. Defaults to 10.
- **auto_font_size** (*bool*) –Whether to automatically adjust font size. Defaults to False.
- **font_families** (*Union[str, List[str]]*) –The font families of labels. Defaults to 'sans-serif'.
- **font_properties** (*Union[str, FontProperties], optional*) –The font properties of texts. The format should be a path str to font file or a *font_manager.FontProperties()* object. If you want to draw Chinese texts, you need to prepare a font file that can show Chinese characters properly. For example: *simhei.ttf*,'simsun.ttc','simkai.ttf' and so on. Then set font_properties=matplotlib.font_manager.FontProperties (fname='path/to/font_file') or font_properties='path/to/font_file'. This function need mmengine version >=0.6.0. Defaults to None.

返回类型 numpy.ndarray

get_polygons_image(*image*, *polygons*, *colors='g'*, *filling=False*, *line_width=0.5*, *alpha=0.5*)

Draw polygons on image.

参数

- **image** (*np.ndarray*) –The origin image to draw. The format should be RGB.
- **polygons** (*Sequence[np.ndarray]*) –The polygons to draw. The shape should be (N, 2).
- **colors** (*Union[str, Sequence[str]]*) –The colors of polygons. `colors` can have the same length with polygons or just single value. If `colors` is single value, all the polygons will have the same colors. Refer to *matplotlib.colors* for full list of formats that are accepted. Defaults to 'g'.
- **filling** (*bool*) –Whether to fill the polygons. Defaults to False.
- **line_width** (*Union[int, float]*) –The line width of polygons. Defaults to 0.5.

- **alpha** (*float*) –The alpha of polygons. Defaults to 0.5.

返回 The image with polygons drawn.

返回类型 np.ndarray

48.2 TextDetLocalVisualizer

class mmocr.visualization.**TextDetLocalVisualizer**(*name='visualizer'*, *image=None*, *with_poly=True*, *with_bbox=False*, *vis_backends=None*, *save_dir=None*, *gt_color='g'*, *gt_ignored_color='b'*, *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'.
- **gt_ignored_color** (*Union[str, tuple, list[str], list[tuple]]*) –The colors of ignored 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 'b'.
- **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

48.3 TextRecogLocalVisualizer

class mmocr.visualization.**TextRecogLocalVisualizer**(*name='visualizer'*, *image=None*, *vis_backends=None*, *save_dir=None*, *gt_color='g'*, *pred_color='r'*, ***kwargs*)

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

48.4 TextSpottingLocalVisualizer

class mmocr.visualization.**TextSpottingLocalVisualizer**(*name='visualizer'*, *font_families='sans-serif'*, *font_properties=None*, ***kwargs*)

参数

- **name** (*str*) –
- **font_families** (*Union[str, List[str]]*) –
- **font_properties** (*Optional[Union[str, matplotlib.font_manager.FontProperties]]*) –

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=0.5*, *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** (`TextSpottingDataSample`, 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

48.5 KIELocalVisualizer

class mmocr.visualization.**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 49

mmocr.engine

mmocr.engine

- *Hooks*

49.1 Hooks

VisualizationHook	Detection Visualization Hook.

49.1.1 VisualizationHook

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*, *backend_args=None*)

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.
- **backend_args** (*dict, optional*) –Instantiates the corresponding file backend. It may contain *backend* key to specify the file backend. If it contains, the file backend corresponding to this value will be used and initialized with the remaining values, otherwise the corresponding file backend will be selected based on the prefix of the file path. Defaults to None.
- **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 50

mmocr.utils

mmocr.utils

50.1 Image Utils

50.2 Box Utils

`bbox2poly`	Converting a bounding box to a polygon.

下页继续

表 1 – 续上页

bbox_center_distance	Calculate the distance between the center points of two bounding boxes.
bbox_diag_distance	Calculate the diagonal length of a bounding box (distance between the top-left and bottom-right).
bezier2polygon	Sample points from the boundary of a polygon enclosed by two Bezier curves, which are controlled by bezier_points.
is_on_same_line	Check if two boxes are on the same line by their y-axis coordinates.
rescale_bboxes	Rescale bboxes according to scale_factor.
stitch_boxes_into_lines	Stitch fragmented boxes of words into lines.

50.2.1 mmocr.utils.bbox2poly

mmocr.utils.**bbox2poly**(*bbox*, *mode='xyxy'*)

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].

- **mode** (*str*) –Specify the format of bbox. Can be ‘xyxy’ or ‘xywh’. Defaults to ‘xyxy’.

返回 The converted polygon [x1, y1, x2, y1, x2, y2, x1, y2].

返回类型 np.array

50.2.2 mmocr.utils.bbox_center_distance

mmocr.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

50.2.3 mmocr.utils.bbox_diag_distance

mmocr.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

50.2.4 mmocr.utils.bezier2polygon

mmocr.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.

50.2.5 mmocr.utils.is_on_same_line

mmocr.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

50.2.6 mmocr.utils.rescale_bboxes

mmocr.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]

50.2.7 mmocr.utils.stitch_boxes_into_lines

mmocr.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])

50.3 Point Utils

point_distance	Calculate the distance between two points.
points_center	Calculate the center of a set of points.

50.3.1 mmocr.utils.point_distance

mmocr.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

50.3.2 mmocr.utils.points_center

mmocr.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

50.4 Polygon Utils

boundary_iou	Calculate the IOU between two boundaries.
crop_polygon	Crop polygon to be within a box region.
is_poly_inside_rect	Check if the polygon is inside the target region.
offset_polygon	Offset (expand/shrink) the polygon by the target distance.
poly2bbox	Converting a polygon to a bounding box.
poly2shapely	Convert a polygon to shapely.geometry.Polygon.
poly_intersection	Calculate the intersection area between two polygons.
poly_iou	Calculate the IOU between two polygons.
poly_make_valid	Convert a potentially invalid polygon to a valid one by eliminating self-crossing or self-touching parts.
poly_union	Calculate the union area between two polygons.
polys2shapely	Convert a nested list of boundaries to a list of Polygons.
rescale_polygon	Rescale a polygon according to scale_factor.
rescale_polygons	Rescale polygons according to scale_factor.
shapely2poly	Convert a nested list of boundaries to a list of Polygons.
sort_points	Sort arbitrary points in clockwise order in Cartesian coordinate, you may need to reverse the output sequence if you are using OpenCV’s image coordinate.
sort_vertex	Sort box vertices in clockwise order from left-top first.
sort_vertex8	Sort vertex with 8 points [x1 y1 x2 y2 x3 y3 x4 y4]

50.4.1 mmocr.utils.boundary_iou

mmocr.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

50.4.2 mmocr.utils.crop_polygon

mmocr.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

50.4.3 mmocr.utils.is_poly_inside_rect

mmocr.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[b int, float, complex, str, bytes]]]]]]) –

- **rect** (*numpy.ndarray*) –

50.4.4 mmocr.utils.offset_polygon

mmocr.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

50.4.5 mmocr.utils.poly2bbox

mmocr.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

50.4.6 mmocr.utils.poly2shapely

mmocr.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)

50.4.7 mmocr.utils.poly_intersection

mmocr.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, which will be *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)

50.4.8 mmocr.utils.poly_iou

mmocr.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

50.4.9 mmocr.utils.poly_make_valid

mmocr.utils.**poly_make_valid**(*poly*)

Convert a potentially invalid polygon to a valid one by eliminating self-crossing or self-touching parts. Note that if the input is a line, the returned polygon could be an empty one.

参数 **poly** (*Polygon*) –A polygon needed to be converted.

返回 A valid polygon, which might be empty.

返回类型 Polygon

50.4.10 mmocr.utils.poly_union

mmocr.utils.**poly_union**(*poly_a*, *poly_b*, *invalid_ret=None*, *return_poly=False*)

Calculate the union 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 False.
- **return_poly** (*bool*) –Whether to return the polygon of the union. Defaults to False.

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

50.4.11 mmocr.utils.polys2shapely

mmocr.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

50.4.12 mmocr.utils.rescale_polygon

mmocr.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

50.4.13 mmocr.utils.rescale_polygons

mmocr.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] or 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. The type of the return value depends on the type of the input polygons.

返回类型 list[np.ndarray] or np.ndarray

50.4.14 mmocr.utils.shapely2poly

mmocr.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

50.4.15 mmocr.utils.sort_points

mmocr.utils.**sort_points**(*points*)

Sort arbitrary points in clockwise order in Cartesian coordinate, you may need to reverse the output sequence if you are using OpenCV's image coordinate.

Reference: https://github.com/novioleo/Savior/blob/master/Utils/GeometryUtils.py.

Warning: This function can only sort convex polygons.

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]

50.4.16 mmocr.utils.sort_vertex

mmocr.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]]

50.4.17 mmocr.utils.sort_vertex8

mmocr.utils.**sort_vertex8**(*points*)

Sort vertex with 8 points [x1 y1 x2 y2 x3 y3 x4 y4]

50.5 Mask Utils

fill_hole	Fill holes in matrix.

50.5.1 mmocr.utils.fill_hole

mmocr.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

50.6 Misc Utils

equal_len	
is_2dlist	check x is 2d-list([[1], []]) or 1d empty list([]).
is_3dlist	check x is 3d-list([[[1], []]]) or 2d empty list([[], []]) or 1d empty list([]).
is_none_or_type	
is_type_list	

50.6.1 mmocr.utils.equal_len

mmocr.utils.**equal_len**(**argv*)

50.6.2 mmocr.utils.is_2dlist

mmocr.utils.**is_2dlist**(*x*)

check x is 2d-list([[1], []]) or 1d empty list([]).

Notice: The reason that it contains 1d empty list is because some arguments from gt annotation file or model prediction may be empty, but usually, it should be 2d-list.

50.6.3 mmocr.utils.is_3dlist

mmocr.utils.**is_3dlist**(*x*)

check x is 3d-list([[[1], []]]) or 2d empty list([[], []]) or 1d empty list([]).

Notice: The reason that it contains 1d or 2d empty list is because some arguments from gt annotation file or model prediction may be empty, but usually, it should be 3d-list.

50.6.4 mmocr.utils.is_none_or_type

mmocr.utils.**is_none_or_type**(*x*, *type*)

50.6.5 mmocr.utils.is_type_list

mmocr.utils.**is_type_list**(*x*, *type*)

50.7 Setup Env

register_all_modules	Register all modules in mmocr into the registries.

50.7.1 mmocr.utils.register_all_modules

mmocr.utils.**register_all_modules**(*init_default_scope=True*)

Register all modules in mmocr into the registries.

参数 **init_default_scope** (*bool*) –Whether initialize the mmocr default scope. When *init_default_scope=True*, the global default scope will be set to *mmocr*, and all registries will build modules from mmocr's registry node. To understand more about the registry, please refer to https://github.com/open-mmlab/mmengine/blob/main/docs/en/tutorials/registry.md Defaults to True.

返回类型 None

CHAPTER 51

欢迎加入 OpenMMLab 社区

扫描下方的二维码可关注 OpenMMLab 团队的 知乎官方账号，加入 OpenMMLab 团队的 官方交流 QQ 群，或通过添加微信“Open 小喵 Lab”加入官方交流微信群，或者加入我们的 Slack 社区

我们会在 OpenMMLab 社区为大家

- 分享 AI 框架的前沿核心技术
- 解读 PyTorch 常用模块源码
- 发布 OpenMMLab 的相关新闻
- 介绍 OpenMMLab 开发的前沿算法
- 获取更高效的问题答疑和意见反馈
- 提供与各行各业开发者充分交流的平台

干货满满，等你来撩，OpenMMLab 社区期待您的加入

CHAPTER 52

English

CHAPTER 53

简体中文

CHAPTER 54

导引

- genindex
- search

Python 模块索引

m

索引

C

D

E

F

G

N

O

P

R

U

V

W

模块