MMOCR

发布 *0.6.3*

OpenMMLab

2023 年 04 月 25 日

开始

1 安装 **3**

1.1 环境依赖 3

1.2 准备环境 3

1.3 安装步骤 4

1.4 自定义安装 6

1.5 对 MMCV 和 MMDetection 的版本依赖 7

2 开始 **9**

2.1 安装 9

2.2 数据集准备 9

2.3 使用预训练模型进行推理 9

2.4 训练 10

2.5 测试 11

3 演示 **13**

3.1 案例一：文本检测 13

3.2 案例二：文本识别 14

3.3 案例三：文本检测 + 识别 14

3.4 案例 4：文本检测 + 识别 + 关键信息提取 15

3.5 API 参数 16

3.6 模型 16

3.7 其他需要注意 17

4 Training **19**

4.1 Training on a Single GPU 19

4.2 Training on Multiple GPUs 20

4.3 Training on Multiple Machines 20

4.4 Training with Slurm 20

4.5 Commonly Used Training Configs . . . 21

5 测试 **23**
5.1 使用单 GPU 进行测试 . . . 23
5.2 使用多 GPU 进行测试 . . . 24
5.3 使用 Slurm 进行测试 . . . 24
5.4 批量测试 . . . 24

6 部署 **27**
6.1 转换至 ONNX (试验性的) . . . 27
6.2 ONNX 转 TensorRT （试验性的） . . . 28
6.3 评估 ONNX 和 TensorRT 模型（试验性的） . . . 29
6.4 结果和模型 . . . 29

7 服务器部署 **31**
7.1 安装 TorchServe . . . 31
7.2 将 MMOCR 模型转换为 TorchServe 模型格式 . . . 31
7.3 启动服务 . . . 32
7.4 4. 测试单张图片推理 . . . 33

8 Learn about Configs **35**
8.1 Modify config through script arguments . . . 35
8.2 Config Name Style . . . 36
8.3 Config Structure . . . 37
8.4 Config File Structure . . . 37
8.5 FAQ . . . 43
8.6 Deprecated train_cfg/test_cfg . . . 44

9 Dataset Types **47**
9.1 Dataset Wrapper . . . 47
9.2 Text Detection . . . 48
9.3 Text Recognition . . . 52

10 KIE: Difference between CloseSet & OpenSet **57**
10.1 CloseSet . . . 57
10.2 OpenSet . . . 58

11 统计数据 **59**
11.1 关键信息提取模型 . . . 59
11.2 命名实体识别模型 . . . 60
11.3 文本检测模型 . . . 60
11.4 文本识别模型 . . . 60

12 Model Architecture Summary **63**

12.1 Text Detection Models 63
12.2 Text Recognition Models 65
12.3 Key Information Extraction Models 68

13 文本检测模型 **69**
13.1 DBNet 69
13.2 DBNetpp 70
13.3 DRRG 71
13.4 FCENet 72
13.5 Mask R-CNN 73
13.6 PANet 74
13.7 PSENet 75
13.8 Textsnake 76

14 文本识别模型 **79**
14.1 ABINet 79
14.2 CRNN 81
14.3 MASTER 82
14.4 NRTR 83
14.5 RobustScanner 84
14.6 SAR 85
14.7 SATRN 87
14.8 SegOCR 88
14.9 CRNN-STN 88

15 关键信息提取模型 **91**
15.1 SDMGR 91

16 命名实体识别模型 **93**
16.1 Bert 93

17 文字检测 **95**
17.1 概览 95
17.2 重要提醒 96
17.3 准备步骤 96

18 文字识别 **101**
18.1 概览 101
18.2 准备步骤 103

19 关键信息提取 **109**
19.1 概览 109
19.2 准备步骤 109

20 命名实体识别（专名识别） **111**
20.1 概览 . . . 111
20.2 准备步骤 . . . 111

21 Useful Tools **113**
21.1 Publish a Model . . . 113
21.2 Convert text recognition dataset to lmdb format . . . 114
21.3 Convert annotations from Labelme . . . 114
21.4 Log Analysis . . . 115

22 Changelog **117**
22.1 0.6.3 (03/11/2022) . . . 117
22.2 0.6.2 (14/10/2022) . . . 118
22.3 0.6.1 (04/08/2022) . . . 120
22.4 0.6.0 (05/05/2022) . . . 122
22.5 0.5.0 (31/03/2022) . . . 127
22.6 New Contributors . . . 136
22.7 v0.4.1 (27/01/2022) . . . 137
22.8 v0.4.0 (15/12/2021) . . . 139
22.9 v0.3.0 (25/8/2021) . . . 144
22.10 v0.2.1 (20/7/2021) . . . 146
22.11 v0.2.0 (18/5/2021) . . . 147
22.12 v0.1.0 (7/4/2021) . . . 149

23 mmocr.apis **151**

24 mmocr.core **153**
24.1 evaluation . . . 153

25 mmocr.utils **155**

26 mmocr.models **157**
26.1 Common Backbones . . . 158
26.2 Text Detection Detectors . . . 158
26.3 Text Detection Heads . . . 158
26.4 Text Detection Necks . . . 158
26.5 Text Detection Losses . . . 158
26.6 Text Detection Postprocessors . . . 158
26.7 Text Recognition Recognizer . . . 158
26.8 Text Recognition Backbones . . . 158
26.9 Text Recognition Necks . . . 158
26.10 Text Recognition Heads . . . 158
26.11 Text Recognition Preprocessors . . . 158
26.12 Text Recognition Backbones . . . 158

26.13 Text Recognition Layers . . . 158
26.14 Text Recognition Convertors . . . 158
26.15 Text Recognition Encoders . . . 158
26.16 Text Recognition Decoders . . . 158
26.17 Text Recognition Fusers . . . 158
26.18 Text Recognition Losses . . . 158
26.19 KIE Extractors . . . 158
26.20 KIE Heads . . . 158
26.21 KIE Losses . . . 158
26.22 NER Encoders . . . 158
26.23 NER Decoders . . . 158
26.24 NER Losses . . . 158

27 mmocr.datasets **159**
27.1 datasets . . . 159
27.2 pipelines . . . 159
27.3 utils . . . 159

28 欢迎加入 OpenMMLab 社区 **161**

29 导引 **163**

您可以在页面左下角切换中英文文档。

CHAPTER 1

安装

1.1 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

1.2 准备环境

注解: 如果你已经在本地安装了 PyTorch，请直接跳转到安装步骤。

第一步下载并安装 Miniconda.

第二步创建并激活一个 conda 环境:

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

第三步依照官方指南，安装 PyTorch。

在 GPU 平台上：

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

在 CPU 平台上：

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

1.3 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。

1.3.1 推荐步骤

第一步使用 MIM 安装 MMCV.

```
pip install -U openmim
mim install mmcv-full
```

第二步将 MMDetection 以依赖库的形式安装。

```
pip install mmdet
```

第三步安装 MMOCR.

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

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

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

```
pip install mmocr
```

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

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

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

查看 `albumentations` 的官方文档以获知详情。

1.3.2 检验

根据安装方式的不同，我们提供了两个可以验证安装正确性的方法。若 MMOCR 的安装无误，你在这一节完成后应当能看到以图片和文字形式表示的识别结果，示意如下：

```
# 识别结果
[{'filename': 'demo_text_det', 'text': ['yther', 'doyt', 'nan', 'heraies', '188790',
↪'cadets', 'army', 'ipioneered', 'and', 'icottages', 'land', 'hall', 'sgardens',
↪'established', 'ithis', 'preformer', 'social', 'octavial', 'hill', 'pm', 'ct', 'lof
↪', 'aborought']}]
```

若从源码安装 MMOCR

在 MMOCR 的目录运行以下命令：

```
python mmocr/utils/ocr.py --det DB_r18 --recog CRNN demo/demo_text_det.jpg --imshow
```

若以包形式安装 MMOCR

第一步下载必要的配置，权重和图片：

```
mim download mmocr --config dbnet_r18_fpnc_1200e_icdar2015 --dest .
mim download mmocr --config crnn_academic_dataset --dest .
wget https://raw.githubusercontent.com/open-mmlab/mmocr/main/demo/demo_text_det.jpg
```

取决于你的网络环境，下载过程可能会持续几十秒或者更长。一切就绪后，当前目录树应当包含以下文件：

```
├── crnn_academic-a723a1c5.pth
├── crnn_academic_dataset.py
├── dbnet_r18_fpnc_1200e_icdar2015.py
├── dbnet_r18_fpnc_sbn_1200e_icdar2015_20210329-ba3ab597.pth
└── demo_text_det.jpg
```

第二步在 Python 解释器中运行以下代码：

```
from mmocr.utils.ocr import MMOCR
ocr = MMOCR(recog='CRNN', recog_ckpt='crnn_academic-a723a1c5.pth', recog_config='crnn_
↪academic_dataset.py', det='DB_r18', det_ckpt='dbnet_r18_fpnc_sbn_1200e_icdar2015_
↪20210329-ba3ab597.pth', det_config='dbnet_r18_fpnc_1200e_icdar2015.py')
ocr.readtext('demo_text_det.jpg', imshow=True)
```

1.4 自定义安装

1.4.1 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）。

1.4.2 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-full -f https://download.openmmlab.com/mmcv/dist/cu113/torch1.10/
↪index.html
```

1.4.3 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：

1.4.4 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
```

1.5 对 MMCV 和 MMDetection 的版本依赖

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

CHAPTER 2

开始

在这个指南中，我们将介绍一些常用的命令，来帮助你熟悉 MMOCR。我们同时还提供了notebook 版本的代码，可以让您快速上手 MMOCR。

2.1 安装

查看安装指南，了解完整步骤。

2.2 数据集准备

MMOCR 支持许多种类数据集，这些数据集根据其相应任务的类型进行分类。可以在以下部分找到它们的准备步骤：检测数据集、识别数据集、*KIE* 数据集和*NER* 数据集。

2.3 使用预训练模型进行推理

下面通过一个简单的命令来演示端到端的识别：

```
python mmocr/utils/ocr.py demo/demo_text_ocr.jpg --print-result --imshow
```

其检测结果将被打印出来，并弹出一个新窗口显示结果。更多示例和完整说明可以在示例中找到。

2.4 训练

2.4.1 小数据集训练

在 tests/data 目录下提供了一个用于训练演示的小数据集，在准备学术数据集之前，它可以演示一个初步的训练。

例如：用 seg 方法和小数据集来训练文本识别任务，

```
python tools/train.py configs/textrecog/seg/seg_r31_1by16_fpnocr_toy_dataset.py --
↪work-dir seg
```

用 sar 方法和小数据集训练文本识别,

```
python tools/train.py configs/textrecog/sar/sar_r31_parallel_decoder_toy_dataset.py --
↪work-dir sar
```

2.4.2 使用学术数据集进行训练

按照说明准备好所需的学术数据集后，最后要检查模型的配置是否将 MMOCR 指向正确的数据集路径。假设在 ICDAR2015 数据集上训练 DBNet, 部分配置如 configs/_base_/det_datasets/icdar2015.py 所示:

```
dataset_type = 'IcdarDataset'
data_root = 'data/icdar2015'
train = dict(
    type=dataset_type,
    ann_file=f'{data_root}/instances_training.json',
    img_prefix=f'{data_root}/imgs',
    pipeline=None)
test = dict(
    type=dataset_type,
    ann_file=f'{data_root}/instances_test.json',
    img_prefix=f'{data_root}/imgs',
    pipeline=None)
train_list = [train]
test_list = [test]
```

这里需要检查数据集路径 data/icdar2015 是否正确. 然后可以启动训练命令:

```
python tools/train.py configs/textdet/dbnet/dbnet_r18_fpnc_1200e_icdar2015.py --work-
↪dir dbnet
```

想要了解完整的训练参数配置可以查看*Training*了解。

2.5 测试

假设我们完成了 DBNet 模型训练，并将最新的模型保存在 dbnet/latest.pth。则可以使用以下命令，及 hmean-iou 指标来评估其在测试集上的性能：

```
python tools/test.py configs/textdet/dbnet/dbnet_r18_fpnc_1200e_icdar2015.py dbnet/
↪latest.pth --eval hmean-iou
```

还可以在线评估预训练模型，命令如下：

```
python tools/test.py configs/textdet/dbnet/dbnet_r18_fpnc_1200e_icdar2015.py https://
↪download.openmmlab.com/mmocr/textdet/dbnet/dbnet_r18_fpnc_sbn_1200e_icdar2015_
↪20210329-ba3ab597.pth --eval hmean-iou
```

有关测试的更多说明，请参阅测试.

CHAPTER 3

演示

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

该 API 可以通过命令行执行，也可以在 python 脚本内调用。在该 API 里，MMOCR 里的所有模型能以独立模块的形式被调用或串联。它还支持将 Tesseract 作为文字检测或识别的一个组件调用。

3.1 案例一：文本检测

注：使用 TextSnake 检测模型对图像上的文本进行检测，结果用 json 格式的文件（默认）导出，并保存可视化的文件。

- 命令行执行：

```
python mmocr/utils/ocr.py demo/demo_text_det.jpg --output demo/ --det TextSnake --
↪recog None --export demo/
```

- Python 调用：

```
from mmocr.utils.ocr import MMOCR

# 导入模型到内存
ocr = MMOCR(det='TextSnake', recog=None)

```

(下页继续)

(续上页)

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

3.2 案例二：文本识别

注： 使用 CRNN_TPS 识别模型对多张图片进行批量识别。*批处理的尺寸设置为 10，以防内存溢出引起的 CUDA 运行时错误。*

- 命令行执行：

```
python mmocr/utils/ocr.py %INPUT_FOLDER_PATH% --det None --recog CRNN_TPS --batch-
↪mode --single-batch-size 10 --output %OUPUT_FOLDER_PATH%
```

- Python 调用：

```
from mmocr.utils.ocr import MMOCR

# 导入模型到内存
ocr = MMOCR(det=None, recog='CRNN_TPS')

# 推理
results = ocr.readtext(%INPUT_FOLDER_PATH%, output = %OUTPUT_FOLDER_PATH%, batch_
↪mode=True, single_batch_size = 10)
```

3.3 案例三：文本检测 + 识别

注： 使用 PANet_IC15（默认）检测模型和 SAR（默认）识别模型，对 demo/demo_text_det.jpg 图片执行 ocr（检测 + 识别）推理，在终端打印结果并展示可视化结果。

- 命令行执行：

```
python mmocr/utils/ocr.py demo/demo_text_ocr.jpg --print-result --imshow
```

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

- Python 调用：

```python
from mmocr.utils.ocr import MMOCR

# 导入模型到内存
ocr = MMOCR()

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

3.4 案例 4：文本检测 + 识别 + 关键信息提取

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

- 命令行执行：

```
python mmocr/utils/ocr.py demo/demo_kie.jpeg  --det PS_CTW --recog SAR --kie SDMGR --
↪print-result --imshow
```

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

- Python 调用：

```python
from mmocr.utils.ocr import MMOCR

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

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

3.5 API 参数

该 API 有多个可供使用的参数列表。下表是 python 接口的参数。

MMOCR():

[1]: `kie` 当且仅当同时指定了文本检测和识别模型时才有效。

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

3.5.1 readtext()

[1]: `batch_mode` 需确保模型兼容批处理模式（见下表模型是否支持批处理）。

[2]: `merge` 只有同时运行检测 + 识别模式，参数才有效。

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

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

3.6 模型

文本检测：

文本识别：

注解： SAR_CN 是唯一支持中文字符识别的模型，并且它需要一个中文字典。以便推理能成功运行，请先从 这里 下载辞典。

关键信息提取：

3.7 其他需要注意

- 执行检测 + 识别的推理（端到端 ocr），需要同时定义 `det` 和 `recog` 参数
- 如果只需要执行检测，则 `recog` 参数设置为 `None`。
- 如果只需要执行识别，则 `det` 参数设置为 `None`。
- `details` 参数仅在端到端的 ocr 模型有效。
- `det_batch_size` 和 `recog_batch_size` 指定了在同时间传递给模型的图片数量。为了提高推理速度，应该尽可能设置你能设置的最大值。最大的批处理值受模型复杂度和 GPU 的显存大小限制。
- MMOCR 目前通过 `tesserocr` 调用 Tesseract 的 API.

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

CHAPTER 4

Training

4.1 Training on a Single GPU

You can use `tools/train.py` to train a model on a single machine with a CPU and optionally a GPU.

Here is the full usage of the script:

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

注解: By default, MMOCR prefers GPU to CPU. If you want to train a model on CPU, please empty `CUDA_VISIBLE_DEVICES` or set it to -1 to make GPU invisible to the program. Note that CPU training requires **MMCV >= 1.4.4**.

```
CUDA_VISIBLE_DEVICES= python tools/train.py ${CONFIG_FILE} [ARGS]
```

4.2 Training on Multiple GPUs

MMOCR implements **distributed** training with `MMDistributedDataParallel`. (Please refer to datasets.md to prepare your datasets)

```
[PORT={PORT}] ./tools/dist_train.sh ${CONFIG_FILE} ${WORK_DIR} ${GPU_NUM} [PY_ARGS]
```

4.3 Training on Multiple Machines

You can launch a task on multiple machines connected to the same network.

```
NNODES=${NNODES} NODE_RANK=${NODE_RANK} PORT=${MASTER_PORT} MASTER_ADDR=${MASTER_ADDR}
↪ ./tools/dist_train.sh ${CONFIG_FILE} ${WORK_DIR} ${GPU_NUM} [PY_ARGS]
```

注解: MMOCR relies on torch.distributed package for distributed training. Find more information at PyTorch’s launch utility.

Say that you want to launch a job on two machines. On the first machine:

```
NNODES=2 NODE_RANK=0 PORT=${MASTER_PORT} MASTER_ADDR=${MASTER_ADDR} ./tools/dist_
↪train.sh ${CONFIG_FILE} ${WORK_DIR} ${GPU_NUM} [PY_ARGS]
```

On the second machine:

```
NNODES=2 NODE_RANK=1 PORT=${MASTER_PORT} MASTER_ADDR=${MASTER_ADDR} ./tools/dist_
↪train.sh ${CONFIG_FILE} ${WORK_DIR} ${GPU_NUM} [PY_ARGS]
```

注解: The speed of the network could be the bottleneck of training.

4.4 Training with Slurm

If you run MMOCR on a cluster managed with Slurm, you can use the script `slurm_train.sh`.

```
[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]
```

Here is an example of using 8 GPUs to train a text detection model on the dev partition.

```
./tools/slurm_train.sh dev psenet-ic15 configs/textdet/psenet/psenet_r50_fpnf_sbn_1x_
↪icdar2015.py /nfs/xxxx/psenet-ic15
```

4.4.1 Running Multiple Training Jobs on a Single Machine

If you are launching multiple training jobs on a single machine with Slurm, you may need to modify the port in configs to avoid communication conflicts.

For example, in `config1.py`,

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

In `config2.py`,

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

Then you can launch two jobs with `config1.py` ang `config2.py`.

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

4.5 Commonly Used Training Configs

Here we list some configs that are frequently used during training for quick reference.

```
total_epochs = 1200
data = dict(
    # Note: User can configure general settings of train, val and test dataloader by␣
↪specifying them here. However, their values can be overridden in dataloader's␣
↪config.
    samples_per_gpu=8, # Batch size per GPU
    workers_per_gpu=4, # Number of workers to process data for each GPU
    train_dataloader=dict(samples_per_gpu=10, drop_last=True),   # Batch size = 10,␣
↪workers_per_gpu = 4
    val_dataloader=dict(samples_per_gpu=6, workers_per_gpu=1),  # Batch size = 6,␣
↪workers_per_gpu = 1
    test_dataloader=dict(workers_per_gpu=16),  # Batch size = 8, workers_per_gpu = 16
    ...
)
# Evaluation
```

(下页继续)

(续上页)

```
evaluation = dict(interval=1, by_epoch=True)  # Evaluate the model every epoch
# Saving and Logging
checkpoint_config = dict(interval=1)  # Save a checkpoint every epoch
log_config = dict(
    interval=5,  # Print out the model's performance every 5 iterations
    hooks=[
        dict(type='TextLoggerHook')
    ])
# Optimizer
optimizer = dict(type='SGD', lr=0.02, momentum=0.9, weight_decay=0.0001)  # Supports␣
↪all optimizers in PyTorch and shares the same parameters
optimizer_config = dict(grad_clip=None)  # Parameters for the optimizer hook. See␣
↪https://github.com/open-mmlab/mmcv/blob/master/mmcv/runner/hooks/optimizer.py for␣
↪implementation details
# Learning policy
lr_config = dict(policy='poly', power=0.9, min_lr=1e-7, by_epoch=True)
```

CHAPTER 5

测试

此文档介绍在数据集上测试预训练模型的方法。

5.1 使用单 GPU 进行测试

您可以使用 tools/test.py 执行单 CPU/GPU 推理。例如，要在 IC15 上评估 DBNet:（可以从 Model Zoo 下载预训练模型）:

```
./tools/dist_test.sh configs/textdet/dbnet/dbnet_r18_fpnc_1200e_icdar2015.py dbnet_
↪r18_fpnc_sbn_1200e_icdar2015_20210329-ba3ab597.pth --eval hmean-iou
```

下面是脚本的完整用法:

```
python tools/test.py ${CONFIG_FILE} ${CHECKPOINT_FILE} [ARGS]
```

注解: 默认情况下，MMOCR 更偏向于使用 GPU 而非 CPU。如果您想在 CPU 上测试模型，请清空 CUDA_VISIBLE_DEVICES 或者将其设置为 -1 以使 GPU(s) 对程序不可见。需要注意的是，运行 CPU 测试需要 **MMCV >= 1.4.4**。

```
CUDA_VISIBLE_DEVICES= python tools/test.py ${CONFIG_FILE} ${CHECKPOINT_FILE} [ARGS]
```

5.2 使用多 GPU 进行测试

MMOCR 使用 `MMDistributedDataParallel` 实现 **分布式**测试。

您可以使用以下命令测试具有多个 GPU 的数据集。

```
[PORT={PORT}] ./tools/dist_test.sh ${CONFIG_FILE} ${CHECKPOINT_FILE} ${GPU_NUM} [PY_
↪ARGS]
```

例如，

```
./tools/dist_test.sh configs/example_config.py work_dirs/example_exp/example_model_
↪20200202.pth 1 --eval hmean-iou
```

5.3 使用 Slurm 进行测试

如果您在使用 Slurm 管理的集群上运行 MMOCR，则可以使用脚本 `tools/slurm_test.sh`。

```
[GPUS=${GPUS}] [GPUS_PER_NODE=${GPUS_PER_NODE}] [SRUN_ARGS=${SRUN_ARGS}] ./tools/
↪slurm_test.sh ${PARTITION} ${JOB_NAME} ${CONFIG_FILE} ${CHECKPOINT_FILE} [PY_ARGS]
```

下面是一个在 "dev" 分区上运行任务的示例。该任务名为 "test_job"，其调用了 8 个 GPU 对示例模型进行评估。

```
GPUS=8 ./tools/slurm_test.sh dev test_job configs/example_config.py work_dirs/example_
↪exp/example_model_20200202.pth --eval hmean-iou
```

5.4 批量测试

默认情况下，MMOCR 仅对逐张图像进行测试。为了令推理更快，您可以在配置中更改 `data.val_dataloader.samples_per_gpu` 和 `data.test_dataloader.samples_per_gpu` 字段。

例如，

```
data = dict(
    ...
    val_dataloader=dict(samples_per_gpu=16),
    test_dataloader=dict(samples_per_gpu=16),
    ...
)
```

将使用 16 张图像作为一个批大小测试模型。

警告： 批量测试时数据预处理管道的行为会有所变化，因而可能导致模型的性能下降。

CHAPTER 6

部署

我们在 tools/deployment 目录下提供了一些部署工具。

6.1 转换至 ONNX (试验性的)

我们提供了将模型转换至 ONNX 格式的脚本。转换后的模型可以使用诸如 Netron 的工具可视化。此外，我们也支持比较 PyTorch 和 ONNX 模型的输出结果。

```
python tools/deployment/pytorch2onnx.py
    ${MODEL_CONFIG_PATH} \
    ${MODEL_CKPT_PATH} \
    ${MODEL_TYPE} \
    ${IMAGE_PATH} \
    --output-file ${OUTPUT_FILE} \
    --device-id ${DEVICE_ID} \
    --opset-version ${OPSET_VERSION} \
    --verify \
    --verbose \
    --show \
    --dynamic-export
```

参数说明:

注解: 这个工具仍然是试验性的。一些定制的操作没有被支持，并且我们目前仅支持一部分的文本检测和文

本识别算法。

6.1.1 支持导出到 ONNX 的模型列表

下表列出的模型可以保证导出到 ONNX 并且可以在 ONNX Runtime 下运行。

注解:

- *以上所有模型测试基于 PyTorch==1.8.1，onnxruntime==1.7.0 进行*
- 如果你在上述模型中遇到问题，请创建一个 issue，我们会尽快处理。
- 因为这个特性是试验性的，可能变动很快，请尽量使用最新版的 `mmcv` 和 `mmocr` 尝试。

6.2 ONNX 转 TensorRT （试验性的）

我们也提供了从 ONNX 模型转换至 TensorRT 格式的脚本。另外，我们支持比较 ONNX 和 TensorRT 模型的输出结果。

```
python tools/deployment/onnx2tensorrt.py
    ${MODEL_CONFIG_PATH} \
    ${MODEL_TYPE} \
    ${IMAGE_PATH} \
    ${ONNX_FILE} \
    --trt-file ${OUT_TENSORRT} \
    --max-shape INT INT INT INT \
    --min-shape INT INT INT INT \
    --workspace-size INT \
    --fp16 \
    --verify \
    --show \
    --verbose
```

参数说明：

注解: 这个工具仍然是试验性的。一些定制的操作模型没有被支持。我们目前仅支持一部的文本检测和文本识别算法。

6.2.1 支持导出到 TensorRT 的模型列表

下表列出的模型可以保证导出到 TensorRT 引擎并且可以在 TensorRT 下运行。

注解:

- 以上所有模型测试基于 *PyTorch==1.8.1*，*onnxruntime==1.7.0*，*tensorrt==7.2.1.6* 进行
- 如果你在上述模型中遇到问题，请创建一个 issue，我们会尽快处理。
- 因为这个特性是试验性的，可能变动很快，请尽量使用最新版的 `mmcv` 和 `mmocr` 尝试。

6.3 评估 ONNX 和 TensorRT 模型（试验性的）

我们在 `tools/deployment/deploy_test.py` 中提供了评估 TensorRT 和 ONNX 模型的方法。

6.3.1 前提条件

在评估 ONNX 和 TensorRT 模型之前，首先需要安装 ONNX，ONNXRuntime 和 TensorRT。根据 ONNXRuntime in mmcv 和 TensorRT plugin in mmcv 安装 ONNXRuntime 定制操作和 TensorRT 插件。

6.3.2 使用

```
python tools/deploy_test.py \
    ${CONFIG_FILE} \
    ${MODEL_PATH} \
    ${MODEL_TYPE} \
    ${BACKEND} \
    --eval ${METRICS} \
    --device ${DEVICE}
```

6.3.3 参数说明

6.4 结果和模型

注解:

- TensorRT 上采样（upsample）操作和 PyTorch 有一点不同。对于 DBNet 和 PANet，我们建议把上采样的最近邻 (nearest) 模式代替成双线性 (bilinear) 模式。PANet 的替换处在这里，DBNet 的替换处在这里和这里。如在上表中显示的，带有标记 * 的网络的上采样模式均被改变了。
- 注意到，相比最近邻模式，使用更改后的上采样模式会降低性能。然而，默认网络的权重是通过最近邻模式训练的。为了保持在部署中的最佳性能，建议在训练和 TensorRT 部署中使用双线性模式。
- 所有 ONNX 和 TensorRT 模型都使用数据集上的动态尺寸进行评估，图像根据原始配置文件进行预处理。
- 这个工具仍然是试验性的。一些定制的操作模型没有被支持。并且我们目前仅支持一部分的文本检测和文本识别算法。

CHAPTER 7

服务器部署

MMOCR 预先提供了一些脚本来加速模型部署服务流程。下面快速介绍一些在服务器端通过调用 API 来进行模型推理的必要步骤。

7.1 安装 TorchServe

你可以根据官网步骤来安装 TorchServe 和 torch-model-archiver 两个模块。

7.2 将 MMOCR 模型转换为 TorchServe 模型格式

我们提供了一个便捷的工具可以将任何以 .pth 为后缀的模型转换为以 .mar 结尾的模型来满足 TorchServe 使用要求。

```
python tools/deployment/mmocr2torchserve.py ${CONFIG_FILE} ${CHECKPOINT_FILE} \
--output-folder ${MODEL_STORE} \
--model-name ${MODEL_NAME}
```

注解： ${MODEL_STORE} 必须是文件夹的绝对路径。

例如：

```
python tools/deployment/mmocr2torchserve.py \
  configs/textdet/dbnet/dbnet_r18_fpnc_1200e_icdar2015.py \
  checkpoints/dbnet_r18_fpnc_1200e_icdar2015.pth \
  --output-folder ./checkpoints \
  --model-name dbnet
```

7.3 启动服务

7.3.1 本地启动

准备好模型后，使用一行命令即可启动服务：

```
# 加载所有位于 ./checkpoints 中的模型文件
torchserve --start --model-store ./checkpoints --models all
# 或者你仅仅使用一个模型服务，比如 dbnet
torchserve --start --model-store ./checkpoints --models dbnet=dbnet.mar
```

然后，你可以通过 TorchServe 的 REST API 访问 Inference、Management、Metrics 等服务。你可以在TorchServe REST API 中找到它们的用法。

注解： TorchServe 默认会将服务绑定到端口 8080、8081 、8082 上。你可以通过修改 config.properties 来更改端口及存储位置等内容，并通过可选项 --ts-config config.properties 来运行 TorchServe 服务。

```
inference_address=http://0.0.0.0:8080
management_address=http://0.0.0.0:8081
metrics_address=http://0.0.0.0:8082
number_of_netty_threads=32
job_queue_size=1000
model_store=/home/model-server/model-store
```

7.3.2 通过 Docker 启动

通过 Docker 提供模型服务不失为一种更好的方法。我们提供了一个 Dockerfile，可以让你摆脱那些繁琐且容易出错的环境设置步骤。

构建 mmocr-serve Docker 镜像

```
docker build -t mmocr-serve:latest docker/serve/
```

通过 Docker 运行 mmocr-serve

为了在 GPU 环境下运行 Docker，首先需要安装 nvidia-docker；或者你也可以只使用 CPU 环境而不必加 --gpus 参数。

下面的命令将使用 gpu 运行，将 Inference、Management、Metric 的端口分别绑定到 8080、8081、8082 上，将容器的 IP 绑定到 127.0.0.1 上，并将检查点文件夹 ./checkpoints 从主机挂载到容器的 /home/model-server/model-store 文件夹下。更多相关信息，请查看官方文档中 docker 中运行 TorchServe 服务。

```
docker run --rm \
--cpus 8 \
--gpus device=0 \
-p8080:8080 -p8081:8081 -p8082:8082 \
--mount type=bind,source=`realpath ./checkpoints`,target=/home/model-server/model-
↪store \
mmocr-serve:latest
```

注解： realpath ./checkpoints 指向的是 "./checkpoints" 的绝对路径，你也可以将其替换为你的 torchserve 模型所在的绝对路径。

运行 docker 后，你可以通过 TorchServe 的 REST API 访问 Inference、Management、Metrics 等服务。具体你可以在TorchServe REST API 中找到它们的用法。

7.4 4. 测试单张图片推理

推理 API 允许用户上传一张图到模型服务中，并返回相应的预测结果。

```
curl http://127.0.0.1:8080/predictions/${MODEL_NAME} -T demo/demo_text_det.jpg
```

例如，

```
curl http://127.0.0.1:8080/predictions/dbnet -T demo/demo_text_det.jpg
```

对于检测模型，你会获取到名为 boundary_result 的 json 对象。内部的每个数组包含以浮点数格式的，按顺时针排序的 x，y 边界顶点坐标。数组的最后一位为置信度分数。

```
{
  "boundary_result": [
    [
      221.18990004062653,
      226.875,
      221.18990004062653,
      212.625,
      244.05868631601334,
      212.625,
      244.05868631601334,
      226.875,
      0.80883354575186
    ]
  ]
}
```

对于识别模型，返回的结果如下：

```
{
  "text": "sier",
  "score": 0.5247521847486496
}
```

同时可以使用 `test_torchserve.py` 来可视化对比 TorchServe 和 PyTorch 结果。

```
python tools/deployment/test_torchserve.py ${IMAGE_FILE} ${CONFIG_FILE} ${CHECKPOINT_
↪FILE} ${MODEL_NAME}
[--inference-addr ${INFERENCE_ADDR}] [--device ${DEVICE}]
```

例如：

```
python tools/deployment/test_torchserve.py \
  demo/demo_text_det.jpg \
  configs/textdet/dbnet/dbnet_r18_fpnc_1200e_icdar2015.py \
  checkpoints/dbnet_r18_fpnc_1200e_icdar2015.pth \
  dbnet
```

CHAPTER 8

Learn about Configs

We incorporate modular and inheritance design into our config system, which is convenient to conduct various experiments. If you wish to inspect the config file, you may run `python tools/misc/print_config.py /PATH/TO/CONFIG` to see the complete config.

8.1 Modify config through script arguments

When submitting jobs using "tools/train.py" or "tools/test.py", you may specify `--cfg-options` to in-place modify the config.

- Update config keys of dict chains.

  The config options can be specified following the order of the dict keys in the original config. For example, `--cfg-options model.backbone.norm_eval=False` changes the all BN modules in model backbones to `train` mode.

- Update keys inside a list of configs.

  Some config dicts are composed as a list in your config. For example, the training pipeline `data.train.pipeline` is normally a list e.g. `[dict(type='LoadImageFromFile'), ...]`. If you want to change `'LoadImageFromFile'` to `'LoadImageFromNdarry'` in the pipeline, you may specify `--cfg-options data.train.pipeline.0.type=LoadImageFromNdarry`.

- Update values of list/tuples.

  If the value to be updated is a list or a tuple. For example, the config file normally sets `workflow=[('train', 1)]`. If you want to change this key, you may specify `--cfg-options workflow="[(train,1),(val,` `1)]"`. Note that the quotation mark " is necessary to support list/tuple data types, and that **NO** white space is allowed inside the quotation marks in the specified value.

8.2 Config Name Style

We follow the below style to name full config files (`configs/TASK/*.py`). Contributors are advised to follow the same style.

```
{model}_[ARCHITECTURE]_[schedule]_{dataset}.py
```

`{xxx}` is required field and `[yyy]` is optional.

- `{model}`: model type like `dbnet`, `crnn`, etc.
- `[ARCHITECTURE]`: expands some invoked modules following the order of data flow, and the content depends on the model framework. The following examples show how it is generally expanded.
  - For text detection tasks, key information tasks, and SegOCR in text recognition task: `{model}_[backbone]_[neck]_[schedule]_{dataset}.py`
  - For other text recognition tasks, `{model}_[backbone]_[encoder]_[decoder]_[schedule]_{dataset}.py` Note that `backbone`, `neck`, `encoder`, `decoder` are the names of modules, e.g. `r50`, `fpnocr`, etc.
- `{schedule}`: training schedule. For instance, `1200e` denotes 1200 epochs.
- `{dataset}`: dataset. It can either be the name of a dataset (`icdar2015`), or a collection of datasets for brevity (e.g. `academic` usually refers to a common practice in academia, which uses MJSynth + SynthText as training set, and IIIT5K, SVT, IC13, IC15, SVTP and CT80 as test set).

Most configs are composed of basic *primitive* configs in `configs/_base_`, where each *primitive* config in different subdirectory has a slightly different name style. We present them as follows.

- det_datasets, recog_datasets: `{dataset_name(s)}_[train|test].py`. If [train|test] is not specified, the config should contain both training and test set.

  There are two exceptions: toy_data.py and seg_toy_data.py. In recog_datasets, the first one works for most while the second one contains character level annotations and works for seg baseline only as of Dec 2021.
- det_models, recog_models: `{model}_[ARCHITECTURE].py`.
- det_pipelines, recog_pipelines: `{model}_pipeline.py`.
- schedules: `schedule_{optimizer}_{num_epochs}e.py`.

8.3 Config Structure

For better config reusability, we break many of reusable sections of configs into `configs/_base_`. Now the directory tree of `configs/_base_` is organized as follows:

```
_base_
├── det_datasets
├── det_models
├── det_pipelines
├── recog_datasets
├── recog_models
├── recog_pipelines
└── schedules
```

These *primitive* configs are categorized by their roles in a complete config. Most of model configs are making full use of *primitive* configs by including them as parts of `_base_` section. For example, dbnet_r18_fpnc_1200e_icdar2015.py takes five *primitive* configs from `_base_`:

```
_base_ = [
    '../../_base_/default_runtime.py',
    '../../_base_/schedules/schedule_sgd_1200e.py',
    '../../_base_/det_models/dbnet_r18_fpnc.py',
    '../../_base_/det_datasets/icdar2015.py',
    '../../_base_/det_pipelines/dbnet_pipeline.py'
]
```

From these configs' names we can roughly know this config trains dbnet_r18_fpnc with sgd optimizer in 1200 epochs. It uses the origin dbnet pipeline and icdar2015 as the dataset. We encourage users to follow and take advantage of this convention to organize the config clearly and facilitate fair comparison across different *primitive* configurations as well as models.

Please refer to mmcv for detailed documentation.

8.4 Config File Structure

8.4.1 Model

The parameter `"model"` is a python dictionary in the configuration file, which mainly includes information such as network structure and loss function.

注解: The 'type' in the configuration file is not a constructed parameter, but a class name.

注解: We can also use models from MMDetection by adding `mmdet.` prefix to type name, or from other OpenMMLab projects in a similar way if their backbones are registered in registries.

Shared Section

- `type`: Model name.

Text Detection / Text Recognition / Key Information Extraction Model

- `backbone`: Backbone configs. Common Backbones, TextRecog Backbones
- `neck`: Neck network name. TextDet Necks, TextRecog Necks.
- `bbox_head`: Head network name. Applicable to text detection, key information models and *some* text recognition models. TextDet Heads, TextRecog Heads, KIE Heads.
  - `loss`: Loss function type. TextDet Losses, KIE Losses
  - `postprocessor`: (TextDet only) Postprocess type. TextDet Postprocessors

Text Recognition / Named Entity Extraction Model

- `encoder`: Encoder configs. TextRecog Encoders
- `decoder`: Decoder configs. Applicable to text recognition models. TextRecog Decoders
- `loss`: Loss configs. Applicable to some text recognition models. TextRecog Losses
- `label_convertor`: Convert outputs between text, index and tensor. Applicable to text recognition models. Label Convertors
- `max_seq_len`: The maximum sequence length of recognition results. Applicable to text recognition models.

8.4.2 Data & Pipeline

The parameter `"data"` is a python dictionary in the configuration file, which mainly includes information to construct dataloader:

- `samples_per_gpu` : the BatchSize of each GPU when building the dataloader
- `workers_per_gpu` : the number of threads per GPU when building dataloader
- `train | val | test` : config to construct dataset
  - `type`: Dataset name. Check dataset types for supported datasets.

The parameter `evaluation` is also a dictionary, which is the configuration information of `evaluation hook`, mainly including evaluation interval, evaluation index, etc.

```
# dataset settings
dataset_type = 'IcdarDataset'  # dataset name,
data_root = 'data/icdar2015'  # dataset root
img_norm_cfg = dict(        # Image normalization config to normalize the input images
    mean=[123.675, 116.28, 103.53],  # Mean values used to pre-training the pre-
↪trained backbone models
    std=[58.395, 57.12, 57.375],      # Standard variance used to pre-training the pre-
↪trained backbone models
    to_rgb=True)                      # Whether to invert the color channel, rgb2bgr␣
↪or bgr2rgb.
# train data pipeline
train_pipeline = [  # Training pipeline
    dict(type='LoadImageFromFile'),  # First pipeline to load images from file path
    dict(
        type='LoadAnnotations',  # Second pipeline to load annotations for current␣
↪image
        with_bbox=True,  # Whether to use bounding box, True for detection
        with_mask=True,  # Whether to use instance mask, True for instance␣
↪segmentation
        poly2mask=False),  # Whether to convert the polygon mask to instance mask,␣
↪set False for acceleration and to save memory
    dict(
        type='Resize',  # Augmentation pipeline that resize the images and their␣
↪annotations
        img_scale=(1333, 800),  # The largest scale of image
        keep_ratio=True
    ),  # whether to keep the ratio between height and width.
    dict(
        type='RandomFlip',  # Augmentation pipeline that flip the images and their␣
↪annotations
        flip_ratio=0.5),  # The ratio or probability to flip
    dict(
        type='Normalize',  # Augmentation pipeline that normalize the input images
        mean=[123.675, 116.28, 103.53],  # These keys are the same of img_norm_cfg␣
↪since the
        std=[58.395, 57.12, 57.375],  # keys of img_norm_cfg are used here as␣
↪arguments
        to_rgb=True),
    dict(
        type='Pad',  # Padding config
        size_divisor=32),  # The number the padded images should be divisible
    dict(type='DefaultFormatBundle'),  # Default format bundle to gather data in the␣
↪pipeline
```

(下页继续)

(续上页)

```
    dict(
        type='Collect',  # Pipeline that decides which keys in the data should be␣
↪passed to the detector
        keys=['img', 'gt_bboxes', 'gt_labels', 'gt_masks'])
]
test_pipeline = [
    dict(type='LoadImageFromFile'),  # First pipeline to load images from file path
    dict(
        type='MultiScaleFlipAug',  # An encapsulation that encapsulates the testing␣
↪augmentations
        img_scale=(1333, 800),  # Decides the largest scale for testing, used for the␣
↪Resize pipeline
        flip=False,  # Whether to flip images during testing
        transforms=[
            dict(type='Resize',  # Use resize augmentation
                 keep_ratio=True),  # Whether to keep the ratio between height and␣
↪width, the img_scale set here will be suppressed by the img_scale set above.
            dict(type='RandomFlip'),  # Thought RandomFlip is added in pipeline, it␣
↪is not used because flip=False
            dict(
                type='Normalize',  # Normalization config, the values are from img_
↪norm_cfg
                mean=[123.675, 116.28, 103.53],
                std=[58.395, 57.12, 57.375],
                to_rgb=True),
            dict(
                type='Pad',  # Padding config to pad images divisible by 32.
                size_divisor=32),
            dict(
                type='ImageToTensor',  # convert image to tensor
                keys=['img']),
            dict(
                type='Collect',  # Collect pipeline that collect necessary keys for␣
↪testing.
                keys=['img'])
        ])
]
data = dict(
    samples_per_gpu=32,     # Batch size of a single GPU
    workers_per_gpu=2,      # Worker to pre-fetch data for each single GPU
    train=dict(            # train data config
        type=dataset_type,                 # dataset name
        ann_file=f'{data_root}/instances_training.json',  # Path to annotation file
```

(下页继续)

(续上页)

```
        img_prefix=f'{data_root}/imgs',  # Path to images
        pipeline=train_pipeline),           # train data pipeline
    test=dict(             # test data config
        type=dataset_type,
        ann_file=f'{data_root}/instances_test.json',  # Path to annotation file
        img_prefix=f'{data_root}/imgs',  # Path to images
        pipeline=test_pipeline))
evaluation = dict(       # The config to build the evaluation hook, refer to https://
↪github.com/open-mmlab/mmdetection/blob/master/mmdet/core/evaluation/eval_hooks.py
↪#L7 for more details.
    interval=1,          # Evaluation interval
    metric='hmean-iou')   # Metrics used during evaluation
```

8.4.3 Training Schedule

Mainly include optimizer settings, `optimizer hook` settings, learning rate schedule and `runner` settings:

- `optimizer`: optimizer setting , support all optimizers in `pytorch`, refer to related mmcv documentation.
- `optimizer_config`: `optimizer hook` configuration file, such as setting gradient limit, refer to related mmcv code.
- `lr_config`: Learning rate scheduler, supports "CosineAnnealing", "Step", "Cyclic", etc. Refer to related mmcv documentation for more options.
- `runner`: For `runner`, please refer to `mmcv` for `runner` introduction document.

```
# The configuration file used to build the optimizer, support all optimizers in␣
↪PyTorch.
optimizer = dict(type='SGD',         # Optimizer type
                 lr=0.1,              # Learning rate of optimizers, see detail␣
↪usages of the parameters in the documentation of PyTorch
                 momentum=0.9,        # Momentum
                 weight_decay=0.0001) # Weight decay of SGD
# Config used to build the optimizer hook, refer to https://github.com/open-mmlab/
↪mmcv/blob/master/mmcv/runner/hooks/optimizer.py#L8 for implementation details.
optimizer_config = dict(grad_clip=None)  # Most of the methods do not use gradient␣
↪clip
# Learning rate scheduler config used to register LrUpdater hook
lr_config = dict(policy='step',          # The policy of scheduler, also support␣
↪CosineAnnealing, Cyclic, etc. Refer to details of supported LrUpdater from https://
↪github.com/open-mmlab/mmcv/blob/master/mmcv/runner/hooks/lr_updater.py#L9.
                 step=[30, 60, 90])      # Steps to decay the learning rate
runner = dict(type='EpochBasedRunner',   # Type of runner to use (i.e.␣
↪IterBasedRunner or EpochBasedRunner)
```

(下页继续)

(续上页)

```
              max_epochs=100)     # Runner that runs the workflow in total max_epochs.␣
↪For IterBasedRunner use `max_iters`
```

8.4.4 Runtime Setting

This part mainly includes saving the checkpoint strategy, log configuration, training parameters, breakpoint weight path, working directory, etc..

```
# Config to set the checkpoint hook, Refer to https://github.com/open-mmlab/mmcv/blob/
↪master/mmcv/runner/hooks/checkpoint.py for implementation.
checkpoint_config = dict(interval=1)     # The save interval is 1
# config to register logger hook
log_config = dict(   # Config to register logger hook
    interval=50,  # Interval to print the log
    hooks=[
        dict(type='TextLoggerHook', by_epoch=False),
        dict(type='TensorboardLoggerHook', by_epoch=False),
        dict(type='WandbLoggerHook', by_epoch=False, # The Wandb logger is also␣
↪supported, It requires `wandb` to be installed.
             init_kwargs={
                          'project': "MMOCR", # Project name in WandB
                          }), # Check https://docs.wandb.ai/ref/python/init for more␣
↪init arguments.
        # ClearMLLoggerHook, DvcliveLoggerHook, MlflowLoggerHook, NeptuneLoggerHook,␣
↪PaviLoggerHook, SegmindLoggerHook are also supported based on MMCV implementation.
    ])

dist_params = dict(backend='nccl')    # Parameters to setup distributed training, the␣
↪port can also be set.
log_level = 'INFO'               # The output level of the log.
resume_from = None               # Resume checkpoints from a given path, the training␣
↪will be resumed from the epoch when the checkpoint's is saved.
workflow = [('train', 1)]        # Workflow for runner. [('train', 1)] means there is␣
↪only one workflow and the workflow named 'train' is executed once.
work_dir = 'work_dir'            # Directory to save the model checkpoints and logs for␣
↪the current experiments.
```

8.5 FAQ

8.5.1 Ignore some fields in the base configs

Sometimes, you may set `_delete_=True` to ignore some of fields in base configs. You may refer to mmcv for simple illustration.

8.5.2 Use intermediate variables in configs

Some intermediate variables are used in the configs files, like `train_pipeline`/`test_pipeline` in datasets. It's worth noting that when modifying intermediate variables in the children configs, user need to pass the intermediate variables into corresponding fields again. For example, we usually want the data path to be a variable so that we

```
dataset_type = 'IcdarDataset'
data_root = 'data/icdar2015'

train = dict(
    type=dataset_type,
    ann_file=f'{data_root}/instances_training.json',
    img_prefix=f'{data_root}/imgs',
    pipeline=None)

test = dict(
    type=dataset_type,
    ann_file=f'{data_root}/instances_test.json',
    img_prefix=f'{data_root}/imgs',
    pipeline=None)
```

8.5.3 Use some fields in the base configs

Sometimes, you may refer to some fields in the `_base_` config, so as to avoid duplication of definitions. You can refer to mmcv for some more instructions.

This technique has been widely used in MMOCR's configs, where the main configs refer to the dataset and pipeline defined in *base* configs by:

```
train_list = {{_base_.train_list}}
test_list = {{_base_.test_list}}

train_pipeline = {{_base_.train_pipeline}}
test_pipeline = {{_base_.test_pipeline}}
```

Which assumes that its *base* configs export datasets and pipelines in a way like:

```
#  base dataset config
dataset_type = 'IcdarDataset'
data_root = 'data/icdar2015'

train = dict(
    type=dataset_type,
    ann_file=f'{data_root}/instances_training.json',
    img_prefix=f'{data_root}/imgs',
    pipeline=None)

test = dict(
    type=dataset_type,
    ann_file=f'{data_root}/instances_test.json',
    img_prefix=f'{data_root}/imgs',
    pipeline=None)

train_list = [train]
test_list = [test]
```

```
#  base pipeline config
train_pipeline = dict(...)
test_pipeline = dict(...)
```

8.6 Deprecated train_cfg/test_cfg

The `train_cfg` and `test_cfg` are deprecated in config file, please specify them in the model config. The original config structure is as below.

```
# deprecated
model = dict(
    type=...,
    ...
)
train_cfg=dict(...)
test_cfg=dict(...)
```

The migration example is as below.

```
# recommended
model = dict(
    type=...,
    ...
```

(下页继续)

(续上页)

```
    train_cfg=dict(...),
    test_cfg=dict(...),
)
```

CHAPTER 9

Dataset Types

9.1 Dataset Wrapper

9.1.1 UniformConcatDataset

`UniformConcatDataset` is a fundamental dataset wrapper in MMOCR which allows users to apply a universal pipeline on multiple datasets without specifying the pipeline for each of them.

Applying a Pipeline on Multiple Datasets

For example, to apply `train_pipeline` on both `train1` and `train2`,

```
data = dict(
    ...
    train=dict(
        type='UniformConcatDataset',
        datasets=[train1, train2],
        pipeline=train_pipeline))
```

Also, it support applying different `pipeline` to different `datasets`,

```
train_list1 = [train1, train2]
train_list2 = [train3, train4]
```

(下页继续)

(续上页)

```
data = dict(
    ...
    train=dict(
        type='UniformConcatDataset',
        datasets=[train_list1, train_list2],
        pipeline=[train_pipeline1, train_pipeline2]))
```

Here, `train_pipeline1` will be applied to `train1` and `train2`, and `train_pipeline2` will be applied to `train3` and `train4`.

Getting Mean Evaluation Scores

Evaluating the model on multiple datasets is a common strategy in academia, and the mean score is therefore a critical indicator of the model' s overall performance. By default, `UniformConcatDataset` reports mean scores in the form of `mean_{metric_name}` when more than 1 datasets are wrapped. You can customize the behavior by setting `show_mean_scores` in `data.val` and `data.test`. Choices are `'auto'`(default), `True` and `False`.

```
data = dict(
    ...
    val=dict(
        type='UniformConcatDataset',
        show_mean_scores=True,  # always show mean scores
        datasets=[train_list],
        pipeline=[train_pipeline)
    test=dict(
        type='UniformConcatDataset',
        show_mean_scores=False,  # do not show mean scores
        datasets=[train_list],
        pipeline=[train_pipeline))
```

9.2 Text Detection

9.2.1 IcdarDataset

Dataset with annotation file in coco-like json format

Example Configuration

```
dataset_type = 'IcdarDataset'
prefix = 'tests/data/toy_dataset/'
test=dict(
        type=dataset_type,
        ann_file=prefix + 'instances_test.json',
        img_prefix=prefix + 'imgs',
        pipeline=test_pipeline)
```

Annotation Format

You can check the content of the annotation file in `tests/data/toy_dataset/instances_test.json` for an example. It's compatible with any annotation file in COCO format defined in MMDetection:

注解: Icdar 2015/2017 and ctw1500 annotations need to be converted into the COCO format following the steps in datasets.md.

Evaluation

`IcdarDataset` has implemented two evaluation metrics, `hmean-iou` and `hmean-ic13`, to evaluate the performance of text detection models, where `hmean-iou` is the most widely used metric which computes precision, recall and F-score based on IoU between ground truth and prediction.

In particular, filtering predictions with a reasonable score threshold greatly impacts the performance measurement. MMOCR alleviates such hyperparameter effect by sweeping through the hyperparameter space and returns the best performance every evaluation time. User can tune the searching scheme by passing `min_score_thr`, `max_score_thr` and `step` into the evaluation hook in the config.

For example, with the following configuration, you can evaluate the model's output on a list of boundary score thresholds [0.1, 0.2, 0.3, 0.4, 0.5] and get the best score from them **during training**.

```
evaluation = dict(
    interval=100,
    metric='hmean-iou',
    min_score_thr=0.1,
    max_score_thr=0.5,
    step=0.1)
```

During testing, you can change these parameter values by appending them to `--eval-options`.

```
python tools/test.py configs/textdet/dbnet/dbnet_r18_fpnc_1200e_icdar2015.py db_r18.
↪pth --eval hmean-iou --eval-options min_score_thr=0.1 max_score_thr=0.6 step=0.1
```

Check out our API doc for further explanations on these parameters.

9.2.2 TextDetDataset

Dataset with annotation file in line-json txt format

We have designed new types of dataset consisting of **loader** , **backend**, and **parser** to load and parse different types of annotation files.

- **loader**: Load the annotation file. We now have a unified loader, `AnnFileLoader`, which can use different `backend` to load annotation from txt. The original `HardDiskLoader` and `LmdbLoader` will be deprecated.
- **backend**: Load annotation from different format and backend.
  - `LmdbAnnFileBackend`: Load annotation from lmdb dataset.
  - `HardDiskAnnFileBackend`: Load annotation file with raw hard disks storage backend. The annotation format can be either txt or lmdb.
  - `PetrelAnnFileBackend`: Load annotation file with petrel storage backend. The annotation format can be either txt or lmdb.
  - `HTTPAnnFileBackend`: Load annotation file with http storage backend. The annotation format can be either txt or lmdb.
- **parser**: Parse the annotation file line-by-line and return with `dict` format. There are two types of parser, `LineStrParser` and `LineJsonParser`.
  - `LineStrParser`: Parse one line in ann file while treating it as a string and separating it to several parts by a `separator`. It can be used on tasks with simple annotation files such as text recognition where each line of the annotation files contains the `filename` and `label` attribute only.
  - `LineJsonParser`: Parse one line in ann file while treating it as a json-string and using `json.loads` to convert it to `dict`. It can be used on tasks with complex annotation files such as text detection where each line of the annotation files contains multiple attributes (e.g. `filename`, `height`, `width`, `box`, `segmentation`, `iscrowd`, `category_id`, etc.).

Example Configuration

```
dataset_type = 'TextDetDataset'
img_prefix = 'tests/data/toy_dataset/imgs'
test_anno_file = 'tests/data/toy_dataset/instances_test.txt'
test = dict(
    type=dataset_type,
    img_prefix=img_prefix,
    ann_file=test_anno_file,
    loader=dict(
        type='AnnFileLoader',
        repeat=4,
        parser=dict(
            type='LineJsonParser',
            keys=['file_name', 'height', 'width', 'annotations'])),
    pipeline=test_pipeline,
    test_mode=True)
```

Annotation Format

The results are generated in the same way as the segmentation-based text recognition task above. You can check the content of the annotation file in `tests/data/toy_dataset/instances_test.txt`. The combination of `HardDiskLoader` and `LineJsonParser` will return a dict for each file by calling `__getitem__`:

```
{"file_name": "test/img_10.jpg", "height": 720, "width": 1280, "annotations": [{
↪"iscrowd": 1, "category_id": 1, "bbox": [260.0, 138.0, 24.0, 20.0], "segmentation":␣
↪[[261, 138, 284, 140, 279, 158, 260, 158]]}, {"iscrowd": 0, "category_id": 1, "bbox
↪": [288.0, 138.0, 129.0, 23.0], "segmentation": [[288, 138, 417, 140, 416, 161, 290,
↪ 157]]}, {"iscrowd": 0, "category_id": 1, "bbox": [743.0, 145.0, 37.0, 18.0],
↪"segmentation": [[743, 145, 779, 146, 780, 163, 746, 163]]}, {"iscrowd": 0,
↪"category_id": 1, "bbox": [783.0, 129.0, 50.0, 26.0], "segmentation": [[783, 129,␣
↪831, 132, 833, 155, 785, 153]]}, {"iscrowd": 1, "category_id": 1, "bbox": [831.0,␣
↪133.0, 43.0, 23.0], "segmentation": [[831, 133, 870, 135, 874, 156, 835, 155]]}, {
↪"iscrowd": 1, "category_id": 1, "bbox": [159.0, 204.0, 72.0, 15.0], "segmentation":␣
↪[[159, 205, 230, 204, 231, 218, 159, 219]]}, {"iscrowd": 1, "category_id": 1, "bbox
↪": [785.0, 158.0, 75.0, 21.0], "segmentation": [[785, 158, 856, 158, 860, 178, 787,␣
↪179]]}, {"iscrowd": 1, "category_id": 1, "bbox": [1011.0, 157.0, 68.0, 16.0],
↪"segmentation": [[1011, 157, 1079, 160, 1076, 173, 1011, 170]]}]}
```

Evaluation

TextDetDataset shares a similar implementation with IcdarDataset. Please refer to the evaluation section of '*IcdarDataset*' .

9.3 Text Recognition

9.3.1 OCRDataset

Dataset for encoder-decoder based recognizer

It shares a similar architecture with TextDetDataset. Check out the *introduction* for details.

Example Configuration

```
dataset_type = 'OCRDataset'
img_prefix = 'tests/data/ocr_toy_dataset/imgs'
train_anno_file = 'tests/data/ocr_toy_dataset/label.txt'
train = dict(
    type=dataset_type,
    img_prefix=img_prefix,
    ann_file=train_anno_file,
    loader=dict(
        type='AnnFileLoader',
        repeat=10,
        parser=dict(
            type='LineStrParser',
            keys=['filename', 'text'],
            keys_idx=[0, 1],
            separator=' ')),
    pipeline=train_pipeline,
    test_mode=False)
```

Optional Arguments:

- repeat: The number of repeated lines in the annotation files. For example, if there are 10 lines in the annotation file, setting repeat=10 will generate a corresponding annotation file with size 100.

Annotation Format

You can check the content of the annotation file in `tests/data/ocr_toy_dataset/label.txt`. The combination of `HardDiskLoader` and `LineStrParser` will return a dict for each file by calling `__getitem__`: `{'filename': '1223731.jpg', 'text': 'GRAND'}`.

Loading LMDB Datasets

We have support for reading annotation files from the full lmdb dataset (with images and annotations). It is now possible to read lmdb datasets commonly used in academia. We have also implemented a new dataset conversion tool, recog2lmdb. It converts the recognition dataset to lmdb format. See PR982 for more details.

Here is an example configuration to load lmdb annotations:

```
lmdb_root = 'path to lmdb folder'
train = dict(
    type='OCRDataset',
    img_prefix=lmdb_root,
    ann_file=lmdb_root,
    loader=dict(
        type='AnnFileLoader',
        repeat=1,
        file_format='lmdb',
        parser=dict(
            type='LineJsonParser',
            keys=['filename', 'text']),
    pipeline=None,
    test_mode=False)
```

Evaluation

There are six evaluation metrics available for text recognition tasks: `word_acc`, `word_acc_ignore_case`, `word_acc_ignore_case_symbol`, `char_recall`, `char_precision` and `one_minus_ned`. See our API doc for explanations on metrics.

By default, `OCRDataset` generates full reports on all the metrics if its evaluation metric is `acc`. Here is an example case for **training**.

```
# Configuration
evaluation = dict(interval=1, metric='acc')
```

```
# Results
{'0_char_recall': 0.0484, '0_char_precision': 0.6, '0_word_acc': 0.0, '0_word_acc_
↪ignore_case': 0.0, '0_word_acc_ignore_case_symbol': 0.0, '0_1-N.E.D': 0.0525}
```

注解: ‘0_’ prefixes result from `UniformConcatDataset`. It’ s kept here since MMOCR always wrap `UniformConcatDataset` around any datasets.

If you want to conduct the evaluation on a subset of evaluation metrics:

```
evaluation = dict(interval=1, metric=['word_acc_ignore_case', 'one_minus_ned'])
```

The result will look like:

```
{'0_word_acc_ignore_case': 0.0, '0_1-N.E.D': 0.0525}
```

During testing, you can specify the metrics to evaluate in the command line:

```
python tools/test.py configs/textrecog/crnn/crnn_toy_dataset.py crnn.pth --eval word_
↪acc_ignore_case one_minus_ned
```

9.3.2 OCRSegDataset

Dataset for segmentation-based recognizer

It shares a similar architecture with `TextDetDataset`. Check out the *introduction* for details.

Example Configuration

```
prefix = 'tests/data/ocr_char_ann_toy_dataset/'
train = dict(
    type='OCRSegDataset',
    img_prefix=prefix + 'imgs',
    ann_file=prefix + 'instances_train.txt',
    loader=dict(
        type='AnnFileLoader',
        repeat=10,
        parser=dict(
            type='LineJsonParser',
            keys=['file_name', 'annotations', 'text'])),
    pipeline=train_pipeline,
    test_mode=True)
```

Annotation Format

You can check the content of the annotation file in `tests/data/ocr_char_ann_toy_dataset/instances_train.txt`. The combination of `HardDiskLoader` and `LineJsonParser` will return a dict for each file by calling `__getitem__` each time:

```
{"file_name": "resort_88_101_1.png", "annotations": [{"char_text": "F", "char_box":␣
↪[11.0, 0.0, 22.0, 0.0, 12.0, 12.0, 0.0, 12.0]}, {"char_text": "r", "char_box": [23.
↪0, 2.0, 31.0, 1.0, 24.0, 11.0, 16.0, 11.0]}, {"char_text": "o", "char_box": [33.0,␣
↪2.0, 43.0, 2.0, 36.0, 12.0, 25.0, 12.0]}, {"char_text": "m", "char_box": [46.0, 2.0,
↪ 61.0, 2.0, 53.0, 12.0, 39.0, 12.0]}, {"char_text": ":", "char_box": [61.0, 2.0, 69.
↪0, 2.0, 63.0, 12.0, 55.0, 12.0]}], "text": "From:"}
```

CHAPTER 10

KIE: Difference between CloseSet & OpenSet

Being trained on WildReceipt, SDMG-R, or other KIE models, can identify the types of text boxes on a receipt picture. But what SDMG-R can do is far more beyond that. For example, it' s able to identify key-value pairs on the picture. To demonstrate such ability and hopefully facilitate future research, we release a demonstrative version of WildReceiptOpenset annotated in OpenSet format, and provide a full training/testing pipeline for KIE models such as SDMG-R. Since it might be a *confusing* update, we' ll elaborate on the key differences between the OpenSet and CloseSet format, taking WildReceipt as an example.

10.1 CloseSet

WildReceipt ("CloseSet") divides text boxes into 26 categories. There are 12 key-value pairs of fine-grained key information categories, such as (`Prod_item_value`, `Prod_item_key`), (`Prod_price_value`, `Prod_price_key`) and (`Tax_value`, `Tax_key`), plus two more "do not care" categories: `Ignore` and `Others`.

The objective of CloseSet SDMGR is to predict which category fits the text box best, but it will not predict the relations among text boxes. For instance, if there are four text boxes "Hamburger" , "Hotdog" , "$1" and "$2" on the receipt, the model may assign `Prod_item_value` to the first two boxes and `Prod_price_value` to the last two, but it can' t tell if Hamburger sells for $1 or $2. However, this could be achieved in the open-set variant.

警告: A `*_key` and `*_value` pair do not necessarily have to both appear on the receipt. For example, we usually won' t see `Prod_item_key` appearing on the receipt, while there can be multiple boxes annotated as `Pred_item_value`. In contrast, `Tax_key` and `Tax_value` are likely to appear together since they' re usually structured as `Tax: 11.02` on the receipt.

10.2 OpenSet

In OpenSet, all text boxes, or nodes, have only 4 possible categories: `background`, `key`, `value`, and `others`. The connectivity between nodes are annotated as *edge labels*. If a pair of key-value nodes have the same edge label, they are connected by an valid edge.

Multiple nodes can have the same edge label. However, only key and value nodes will be linked by edges. The nodes of same category will never be connected.

When making OpenSet annotations, each node must have an edge label. It should be an unique one if it falls into non-`key` non-`value` categories.

注解: You can merge `background` to `others` if telling background apart is not important, and we provide this choice in the conversion script for WildReceipt .

10.2.1 Converting WildReceipt from CloseSet to OpenSet

We provide a *conversion script* that converts WildRecipt-like dataset to OpenSet format. This script links every `key-value` pairs following the rules above. Here’s an example illustration: (For better understanding, all the node labels are presented as texts)

警告: A common request from our community is to extract the relations between food items and food prices. In this case, this conversion script ***is not you need***. Wildrecipt doesn’t provide necessary information to recover this relation. For instance, there are four text boxes “Hamburger”, “Hotdog”, “$1” and “$2” on the receipt, and here’s how they actually look like before and after the conversion:

So there won’t be any valid edges connecting them. Nevertheless, OpenSet format is far more general than CloseSet, so this task can be achieved by annotating the data from scratch.

CHAPTER 11

统计数据

- 模型权重文件数量：33
- 配置文件数量：26
- 论文数量：19
  - ALGORITHM: 19

11.1 关键信息提取模型

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

11.2 命名实体识别模型

- 模型权重文件数量：1
- 配置文件数量：1
- 论文数量：1
  - [ALGORITHM] Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding

11.3 文本检测模型

- 模型权重文件数量：15
- 配置文件数量：11
- 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

11.4 文本识别模型

- 模型权重文件数量：14
- 配置文件数量：11
- 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] 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] Robust Scene Text Recognition With Automatic Rectification
- [ALGORITHM] Robustscanner: Dynamically Enhancing Positional Clues for Robust Text Recognition
- [ALGORITHM] Segocr Simple Baseline.
- [ALGORITHM] Show, Attend and Read: A Simple and Strong Baseline for Irregular Text Recognition
- [ALGORITHM] {Master

CHAPTER 12

Model Architecture Summary

MMOCR has implemented many models that support various tasks. Depending on the type of tasks, these models have different architectural designs and, therefore, might be a bit confusing for beginners to master. We release a primary design doc to clearly illustrate the basic task-specific architectures and provide quick pointers to docstrings of model components to aid users' understanding.

12.1 Text Detection Models

The design of text detectors is similar to SingleStageDetector in MMDetection. The feature of an image was first extracted by `backbone` (e.g., ResNet), and `neck` further processes raw features into a head-ready format, where the models in MMOCR usually adapt the variants of FPN to extract finer-grained multi-level features. `bbox_head` is the core of text detectors, and its implementation varies in different models.

When training, the output of `bbox_head` is directly fed into the `loss` module, which compares the output with the ground truth and generates a loss dictionary for optimizer's use. When testing, `Postprocessor` converts the outputs from `bbox_head` to bounding boxes, which will be used for evaluation metrics (e.g., hmean-iou) and visualization.

12.1.1 DBNet

- Backbone: mmdet.ResNet
- Neck: FPNC
- Bbox_head: DBHead
- Loss: DBLoss
- Postprocessor: DBPostprocessor

12.1.2 DRRG

- Backbone: mmdet.ResNet
- Neck: FPN_UNet
- Bbox_head: DRRGHead
- Loss: DRRGLoss
- Postprocessor: DRRGPostprocessor

12.1.3 FCENet

- Backbone: mmdet.ResNet
- Neck: mmdet.FPN
- Bbox_head: FCEHead
- Loss: FCELoss
- Postprocessor: FCEPostprocessor

12.1.4 Mask R-CNN

We use the same architecture as in MMDetection. See MMDetection’ s config documentation for details.

12.1.5 PANet

- Backbone: mmdet.ResNet
- Neck: FPEM_FFM
- Bbox_head: PANHead
- Loss: PANLoss

- Postprocessor: PANPostprocessor

12.1.6 PSENet

- Backbone: mmdet.ResNet
- Neck: FPNF
- Bbox_head: PSEHead
- Loss: PSELoss
- Postprocessor: PSEPostprocessor

12.1.7 Textsnake

- Backbone: mmdet.ResNet
- Neck: FPN_UNet
- Bbox_head: TextSnakeHead
- Loss: TextSnakeLoss
- Postprocessor: TextSnakePostprocessor

12.2 Text Recognition Models

Most of the implemented recognizers use the following architecture:

`preprocessor` refers to any network that processes images before they are fed to `backbone`. `encoder` encodes images features into a hidden vector, which is then transcribed into text tokens by `decoder`.

The architecture diverges at training and test phases. The loss module returns a dictionary during training. In testing, `converter` is invoked to convert raw features into texts, which are wrapped into a dictionary together with confidence scores. Users can access the dictionary with the `text` and `score` keys to query the recognition result.

12.2.1 ABINet

- Preprocessor: None
- Backbone: ResNetABI
- Encoder: ABIVisionModel
- Decoder: ABIVisionDecoder
- Fuser: ABIFuser

- Loss: ABILoss
- Converter: ABIConvertor

注解: Fuser fuses the feature output from encoder and decoder before generating the final text outputs and computing the loss in full ABINet.

12.2.2 CRNN

- Preprocessor: None
- Backbone: VeryDeepVgg
- Encoder: None
- Decoder: CRNNDecoder
- Loss: CTCLoss
- Converter: CTCConvertor

12.2.3 CRNN with TPS-based STN

- Preprocessor: TPSPreprocessor
- Backbone: VeryDeepVgg
- Encoder: None
- Decoder: CRNNDecoder
- Loss: CTCLoss
- Converter: CTCConvertor

12.2.4 MASTER

- Preprocessor: None
- Backbone: ResNet
- Encoder: None
- Decoder: MasterDecoder
- Loss: TFLoss
- Converter: AttnConvertor

12.2.5 NRTR

- Preprocessor: None
- Backbone: ResNet31OCR
- Encoder: NRTREncoder
- Decoder: NRTRDecoder
- Loss: TFLoss
- Converter: AttnConvertor

12.2.6 RobustScanner

- Preprocessor: None
- Backbone: ResNet31OCR
- Encoder: ChannelReductionEncoder
- Decoder: ChannelReductionEncoder
- Loss: SARLoss
- Converter: AttnConvertor

12.2.7 SAR

- Preprocessor: None
- Backbone: ResNet31OCR
- Encoder: SAREncoder
- Decoder: ParallelSARDecoder
- Loss: SARLoss
- Converter: AttnConvertor

12.2.8 SATRN

- Preprocessor: None
- Backbone: ShallowCNN
- Encoder: SatrnEncoder
- Decoder: NRTRDecoder

- Loss: TFLoss
- Converter: AttnConvertor

12.2.9 SegOCR

- Backbone: ResNet31OCR
- Neck: FPNOCR
- Head: SegHead
- Loss: SegLoss
- Converter: SegConvertor

注解: SegOCR' s architecture is an exception - it is closer to text detection models.

12.3 Key Information Extraction Models

The architecture of key information extraction (KIE) models is similar to text detection models, except for the extra feature extractor. As a downstream task of OCR, KIE models are required to run with bounding box annotations indicating the locations of text instances, from which an ROI extractor extracts the cropped features for `bbox_head` to discover relations among them.

The output containing edges and nodes information from `bbox_head` is sufficient for test and inference. Computation of loss also relies on such information.

12.3.1 SDMGR

- Backbone: UNet
- Neck: None
- Extractor: mmdet.SingleRoIExtractor
- Bbox_head: SDMGRHead
- Loss: SDMGRLoss

CHAPTER 13

文本检测模型

13.1 DBNet

Real-time Scene Text Detection with Differentiable Binarization

13.1.1 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.

13.1.2 Results and models

ICDAR2015

13.1.3 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}}
```

13.2 DBNetpp

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

13.2.1 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.

13.2.2 Results and models

ICDAR2015

13.2.3 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}
}
```

13.3 DRRG

Deep relational reasoning graph network for arbitrary shape text detection

13.3.1 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.

13.3.2 Results and models

CTW1500

注解: We’ve upgraded our IoU backend from `Polygon3` to `shapely`. There are some performance differences for some models due to the backends’ different logics to handle invalid polygons (more info here). **New evaluation result is presented in brackets** and new logs will be uploaded soon.

13.3.3 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}
}
```

13.4 FCENet

Fourier Contour Embedding for Arbitrary-Shaped Text Detection

13.4.1 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.

13.4.2 Results and models

CTW1500

ICDAR2015

13.4.3 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}
      }
```

13.5 Mask R-CNN

Mask R-CNN

13.5.1 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.

13.5.2 Results and models

CTW1500

ICDAR2015

ICDAR2017

注解： We tuned parameters with the techniques in Pyramid Mask Text Detector

13.5.3 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}}
```

13.6 PANet

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

13.6.1 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.

13.6.2 Results and models

CTW1500

ICDAR2015

注解: We’ve upgraded our IoU backend from `Polygon3` to `shapely`. There are some performance differences for some models due to the backends’ different logics to handle invalid polygons (more info here). **New evaluation result**

is presented in brackets and new logs will be uploaded soon.

13.6.3 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}
  }
```

13.7 PSENet

Shape robust text detection with progressive scale expansion network

13.7.1 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.

13.7.2 Results and models

CTW1500

ICDAR2015

注解: We' ve upgraded our IoU backend from `Polygon3` to `shapely`. There are some performance differences for some models due to the backends' different logics to handle invalid polygons (more info here). **New evaluation result is presented in brackets** and new logs will be uploaded soon.

13.7.3 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}
}
```

13.8 Textsnake

TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes

13.8.1 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.

13.8.2 Results and models

CTW1500

13.8.3 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 14

文本识别模型

14.1 ABINet

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

14.1.1 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.

14.1.2 Dataset

Train Dataset

Test Dataset

14.1.3 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.
3. Due to some technical issues, the training process of ABINet was interrupted at the 13th epoch and we resumed it later. Both logs are released for full reference.
4. The model architecture in the logs looks slightly different from the final released version, since it was refactored afterward. However, both architectures are essentially equivalent.

14.1.4 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}
}
```

14.2 CRNN

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

14.2.1 Abstract

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

14.2.2 Dataset

Train Dataset

Test Dataset

14.2.3 Results and models

14.2.4 Citation

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

14.3 MASTER

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

14.3.1 Abstract

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

14.3.2 Dataset

Train Dataset

Test Dataset

14.3.3 Results and Models

14.3.4 Citation

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

14.4 NRTR

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

14.4.1 Abstract

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

14.4.2 Dataset

Train Dataset

Test Dataset

14.4.3 Results and Models

注解:

- For backbone `R31-1/16-1/8`:
  - The output consists of 92 classes, including 26 lowercase letters, 26 uppercase letters, 28 symbols, 10 digital numbers, 1 unknown token and 1 end-of-sequence token.
  - The encoder-block number is 6.
  - `1/16-1/8` means the height of feature from backbone is 1/16 of input image, where 1/8 for width.
- For backbone `R31-1/8-1/4`:
  - The output consists of 92 classes, including 26 lowercase letters, 26 uppercase letters, 28 symbols, 10 digital numbers, 1 unknown token and 1 end-of-sequence token.
  - The encoder-block number is 6.

- `1/8-1/4` means the height of feature from backbone is 1/8 of input image, where 1/4 for width.

14.4.4 Citation

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

14.5 RobustScanner

RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

14.5.1 Abstract

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

14.5.2 Dataset

Train Dataset

Test Dataset

14.5.3 Results and Models

14.5.4 References

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

14.5.5 Citation

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

14.6 SAR

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

14.6.1 Abstract

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

14.6.2 Dataset

Train Dataset

Test Dataset

14.6.3 Results and Models

14.6.4 Chinese Dataset

14.6.5 Results and Models

注解:

- `R31-1/8-1/4` means the height of feature from backbone is 1/8 of input image, where 1/4 for width.
- We did not use beam search during decoding.
- We implemented two kinds of decoder. Namely, `ParallelSARDecoder` and `SequentialSARDecoder`.
  - `ParallelSARDecoder`: Parallel decoding during training with `LSTM` layer. It would be faster.
  - `SequentialSARDecoder`: Sequential Decoding during training with `LSTMCell`. It would be easier to understand.
- For train dataset.
  - We did not construct distinct data groups (20 groups in [1]) to train the model group-by-group since it would render model training too complicated.
  - Instead, we randomly selected `2.4m` patches from `Syn90k`, `2.4m` from `SynthText` and `1.2m` from `SynthAdd`, and grouped all data together. See config for details.
- We used 48 GPUs with `total_batch_size = 64 * 48` in the experiment above to speedup training, while keeping the `initial lr = 1e-3` unchanged.

14.6.6 Citation

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

(下页继续)

(续上页)

```
  pages={8610--8617},
  year={2019}
}
```

14.7 SATRN

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

14.7.1 Abstract

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

14.7.2 Dataset

Train Dataset

Test Dataset

14.7.3 Results and Models

14.7.4 Citation

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

14.8 SegOCR

14.8.1 Abstract

Just a simple Seg-based baseline for text recognition tasks.

14.8.2 Dataset

Train Dataset

Test Dataset

14.8.3 Results and Models

注解:

- `R31-1/16` means the size (both height and width) of feature from backbone is 1/16 of input image.
- `1x` means the size (both height and width) of feature from head is the same with input image.

14.8.4 Citation

```
@unpublished{key,
  title={SegOCR Simple Baseline.},
  author={},
  note={Unpublished Manuscript},
  year={2021}
}
```

14.9 CRNN-STN

14.9.1 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.

注解: We use STN from this paper as the preprocessor and CRNN as the recognition network.

14.9.2 Dataset

Train Dataset

Test Dataset

14.9.3 Results and models

14.9.4 Citation

```
@article{shi2016robust,
  title={Robust Scene Text Recognition with Automatic Rectification},
  author={Shi, Baoguang and Wang, Xinggang and Lyu, Pengyuan and Yao,
  Cong and Bai, Xiang},
  year={2016}
}
```

CHAPTER 15

关键信息提取模型

15.1 SDMGR

Spatial Dual-Modality Graph Reasoning for Key Information Extraction

15.1.1 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.

15.1.2 Results and models

WildReceipt

注解:

1. For `sdmgr_novisual`, images are not needed for training and testing. So fake `img_prefix` can be used in configs. As well, fake `file_name` can be used in annotation files.

WildReceiptOpenset

注解:

1. In the case of openset, the number of node categories is unknown or unfixed, and more node category can be added.
2. To show that our method can handle openset problem, we modify the ground truth of `WildReceipt` to `WildReceiptOpenset`. The `nodes` are just classified into 4 classes: `background, key, value, others`, while adding `edge` labels for each box.
3. The model is used to predict whether two nodes are a pair connecting by a valid edge.
4. You can learn more about the key differences between CloseSet and OpenSet annotations in our *tutorial*.

15.1.3 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 16

命名实体识别模型

16.1 Bert

Bert: Pre-training of deep bidirectional transformers for language understanding

16.1.1 Abstract

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

16.1.2 Dataset

Train Dataset

Test Dataset

16.1.3 Results and models

16.1.4 Citation

```
@article{devlin2018bert,
  title={Bert: Pre-training of deep bidirectional transformers for language␣
↪understanding},
  author={Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina},
  journal={arXiv preprint arXiv:1810.04805},
  year={2018}
}
```

CHAPTER 17

文字检测

17.1 概览

文字检测任务的数据集应按如下目录配置:

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

(下页继续)

(续上页)

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

17.2 重要提醒

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

17.3 准备步骤

17.3.1 ICDAR 2015

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

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

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

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

17.3.2 ICDAR 2017

- 与上述步骤类似。

17.3.3 CTW1500

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

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

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

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

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

17.3.4 SynthText

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

17.3.5 TextOCR

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

```
mkdir textocr && cd textocr

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

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

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

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

17.3.6 Totaltext

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

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

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

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

(下页继续)

(续上页)

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

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

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

CHAPTER 18

文字识别

18.1 概览

文字识别任务的数据集应按如下目录配置：

```
├── mixture
│   ├── coco_text
│   │   ├── train_label.txt
│   │   ├── train_words
│   ├── icdar_2011
│   │   ├── training_label.txt
│   │   ├── Challenge1_Training_Task3_Images_GT
│   ├── icdar_2013
│   │   ├── train_label.txt
│   │   ├── test_label_1015.txt
│   │   ├── test_label_1095.txt
│   │   ├── Challenge2_Training_Task3_Images_GT
│   │   ├── Challenge2_Test_Task3_Images
│   ├── icdar_2015
│   │   ├── train_label.txt
│   │   ├── test_label.txt
│   │   ├── ch4_training_word_images_gt
│   │   ├── ch4_test_word_images_gt
│   ├── III5K
│   │   ├── train_label.txt
```

(下页继续)

(续上页)

```
│   │   ├── test_label.txt
│   │   ├── train
│   │   ├── test
│   ├── ct80
│   │   ├── test_label.txt
│   │   ├── image
│   ├── svt
│   │   ├── test_label.txt
│   │   ├── image
│   ├── svtp
│   │   ├── test_label.txt
│   │   ├── image
│   ├── Syn90k
│   │   ├── shuffle_labels.txt
│   │   ├── label.txt
│   │   ├── label.lmdb
│   │   ├── mnt
│   ├── SynthText
│   │   ├── alphanumeric_labels.txt
│   │   ├── shuffle_labels.txt
│   │   ├── instances_train.txt
│   │   ├── label.txt
│   │   ├── label.lmdb
│   │   ├── synthtext
│   ├── SynthAdd
│   │   ├── label.txt
│   │   ├── label.lmdb
│   │   ├── SynthText_Add
│   ├── TextOCR
│   │   ├── image
│   │   ├── train_label.txt
│   │   ├── val_label.txt
│   ├── Totaltext
│   │   ├── imgs
│   │   ├── annotations
│   │   ├── train_label.txt
│   │   ├── test_label.txt
│   ├── OpenVINO
│   │   ├── image_1
│   │   ├── image_2
│   │   ├── image_5
│   │   ├── image_f
│   │   ├── image_val
```

(下页继续)

(续上页)

```
│   │   ├── train_1_label.txt
│   │   ├── train_2_label.txt
│   │   ├── train_5_label.txt
│   │   ├── train_f_label.txt
│   │   ├── val_label.txt
```

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

18.2 准备步骤

18.2.1 ICDAR 2013

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

18.2.2 ICDAR 2015

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

18.2.3 IIIT5K

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

18.2.4 svt

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

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

18.2.5 ct80

- 第一步：下载 test_label.txt

18.2.6 svtp

- 第一步：下载 test_label.txt

18.2.7 coco_text

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

18.2.8 MJSynth (Syn90k)

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

```
mkdir Syn90k && cd Syn90k

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

tar -xzf mjsynth.tar.gz

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

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

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

18.2.9 SynthText (Synth800k)

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

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

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

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

- 第四步：生成裁剪后的图像和标注：

```
cd /path/to/mmocr

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

18.2.10 SynthAdd

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

```
mkdir SynthAdd && cd SynthAdd

mv /path/to/SynthText_Add.zip .

unzip SynthText_Add.zip
```

(下页继续)

(续上页)

```
mv /path/to/label.txt .

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

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

小技巧: 运行以下命令，可以把 `.txt` 格式的标注文件转换成 `.lmdb` 格式:

```
python tools/data/utils/txt2lmdb.py -i <txt_label_path> -o <lmdb_label_path>
```

例如:

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

18.2.11 TextOCR

- 第一步: 下载 train_val_images.zip, TextOCR_0.1_train.json 和 TextOCR_0.1_val.json 到 `textocr/` 目录.

```
mkdir textocr && cd textocr

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

# 对于数据图像
unzip -q train_val_images.zip
mv train_images train
```

- 第二步: 用四个并行进程剪裁图像然后生成 `train_label.txt`, `val_label.txt` , 可以使用以下命令:

```
python tools/data/textrecog/textocr_converter.py /path/to/textocr 4
```

18.2.12 Totaltext

- 第一步：从 github dataset 下载 `totaltext.zip`，然后从 github Groundtruth 下载 `groundtruth_text.zip`（我们建议下载 `.mat` 格式的标注文件，因为我们提供的 `totaltext_converter.py` 标注格式转换工具只支持 `.mat` 文件）

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

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

# 对于标注文件
unzip groundtruth_text.zip
cd Groundtruth
mv Polygon/Train ../annotations/training
mv Polygon/Test ../annotations/test
```

- 第二步：用以下命令生成经剪裁后的标注文件 `train_label.txt` 和 `test_label.txt`（剪裁后的图像会被保存在目录 `data/totaltext/dst_imgs/`）：

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

18.2.13 OpenVINO

- 第零步：安装 awscli。
- 第一步：下载 Open Images 的子数据集 `train_1`、`train_2`、`train_5`、`train_f` 及 `validation` 至 `openvino/`。

```
mkdir openvino && cd openvino

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

# 下载标注文件
for s in 1 2 5 f; do
```

(下页继续)

(续上页)

```
  wget https://storage.openvinotoolkit.org/repositories/openvino_training_extensions/
↪datasets/open_images_v5_text/text_spotting_openimages_v5_train_${s}.json
done
wget https://storage.openvinotoolkit.org/repositories/openvino_training_extensions/
↪datasets/open_images_v5_text/text_spotting_openimages_v5_validation.json

# 解压数据集
mkdir -p openimages_v5/val
for s in 1 2 5 f; do
  tar zxf train_${s}.tar.gz -C openimages_v5
done
tar zxf validation.tar.gz -C openimages_v5/val
```

- 第二步：运行以下的命令，以用 4 个进程生成标注 `train_{1,2,5,f}_label.txt` 和 `val_label.txt` 并裁剪原图：

```
python tools/data/textrecog/openvino_converter.py /path/to/openvino 4
```

CHAPTER 19

关键信息提取

19.1 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
```

19.2 准备步骤

19.2.1 WildReceipt

- 下载并解压 wildreceipt.tar

19.2.2 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 20

命名实体识别（专名识别）

20.1 概览

命名实体识别任务的数据集，文件目录应按如下配置：

```
└── cluener2020
    ├── cluener_predict.json
    ├── dev.json
    ├── README.md
    ├── test.json
    ├── train.json
    └── vocab.txt
```

20.2 准备步骤

20.2.1 CLUENER2020

- 下载并解压 cluener_public.zip 至 `cluener2020/`。
- 下载 vocab.txt 然后将 `vocab.txt` 移动到 `cluener2020/` 文件夹下

CHAPTER 21

Useful Tools

We provide some useful tools under `mmocr/tools` directory.

21.1 Publish a Model

Before you upload a model to AWS, you may want to (1) convert the model weights to CPU tensors, (2) delete the optimizer states and (3) compute the hash of the checkpoint file and append the hash id to the filename. These functionalities could be achieved by `tools/publish_model.py`.

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

For example,

```
python tools/publish_model.py work_dirs/psenet/latest.pth psenet_r50_fpnf_sbn_1x_
↪20190801.pth
```

The final output filename will be `psenet_r50_fpnf_sbn_1x_20190801-{hash id}.pth`.

21.2 Convert text recognition dataset to lmdb format

Reading images or labels from files can be slow when data are excessive, e.g. on a scale of millions. Besides, in academia, most of the scene text recognition datasets are stored in lmdb format, including images and labels. To get closer to the mainstream practice and enhance the data storage efficiency, MMOCR now provides `tools/data/utils/lmdb_converter.py` to convert text recognition datasets to lmdb format.

21.2.1 Examples

Generate a mixed lmdb file with label.txt and images in `imgs/`:

```
python tools/data/utils/lmdb_converter.py label.txt imgs.lmdb -i imgs
```

Generate a mixed lmdb file with label.jsonl and images in `imgs/`:

```
python tools/data/utils/lmdb_converter.py label.json imgs.lmdb -i imgs -f jsonl
```

Generate a label-only lmdb file with label.txt:

```
python tools/data/utils/lmdb_converter.py label.txt label.lmdb --label-only
```

Generate a label-only lmdb file with label.jsonl:

```
python tools/data/utils/lmdb_converter.py label.json label.lmdb --label-only -f jsonl
```

21.3 Convert annotations from Labelme

Labelme is a popular graphical image annotation tool. You can convert the labels generated by labelme to the MMOCR data format using `tools/data/common/labelme_converter.py`. Both detection and recognition tasks are supported.

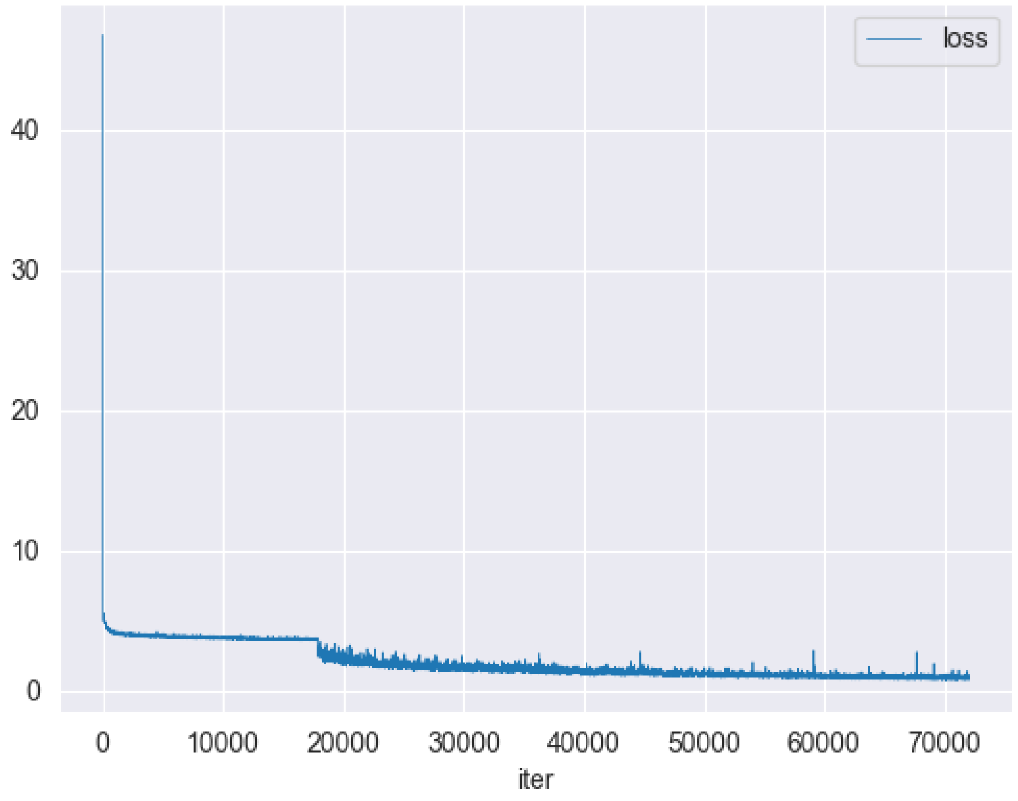
```
# tasks can be "det" or both "det", "recog"
python tools/data/common/labelme_converter.py <json_dir> <image_dir> <out_dir> --
↪tasks <tasks>
```

For example, converting the labelme format annotation in `tests/data/toy_dataset/labelme` to MMOCR detection labels `instances_training.txt` and cropping the image patches for recognition task to `tests/data/toy_dataset/crops` with the labels `train_label.jsonl`:

```
python tools/data/common/labelme_converter.py tests/data/toy_dataset/labelme tests/
↪data/toy_dataset/imgs tests/data/toy_dataset --tasks det recog
```

21.4 Log Analysis

You can use `tools/analyze_logs.py` to plot loss/hmean curves given a training log file. Run `pip install seaborn` first to install the dependency.



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

Examples:

Download the following DBNet and CRNN training logs to run demos.

```
wget https://download.openmmlab.com/mmocr/textdet/dbnet/dbnet_r18_fpnc_sbn_1200e_
↪icdar2015_20210329-ba3ab597.log.json -O DBNet_log.json

wget https://download.openmmlab.com/mmocr/textrecog/crnn/20210326_111035.log.json -O␣
↪CRNN_log.json
```

Please specify an output path if you are running the codes on systems without a GUI.

- Plot loss metric.

```
python tools/analyze_logs.py plot_curve DBNet_log.json --keys loss --legend loss
```

- Plot hmean-iou:hmean metric of text detection.

```
python tools/analyze_logs.py plot_curve DBNet_log.json --keys hmean-iou:hmean --
↪legend hmean-iou:hmean
```

- Plot 0_1-N.E.D metric of text recognition.

```
python tools/analyze_logs.py plot_curve CRNN_log.json --keys 0_1-N.E.D --legend 0_
↪1-N.E.D
```

- Compute the average training speed.

```
python tools/analyze_logs.py cal_train_time CRNN_log.json --include-outliers
```

The output is expected to be like the following.

```
-----Analyze train time of CRNN_log.json-----
slowest epoch 4, average time is 0.3464
fastest epoch 5, average time is 0.2365
time std over epochs is 0.0356
average iter time: 0.2906 s/iter
```

CHAPTER 22

Changelog

22.1 0.6.3 (03/11/2022)

22.1.1 Highlights

This release enhances the inference script and fixes a bug that might cause failure on TorchServe.

Besides, a new backbone, oCLIP-ResNet, and a dataset preparation tool, Dataset Preparer, have been released in MMOCR 1.0.0rc3 (1.x branch). Check out the changelog for more information about the features, and maintenance plan for how we will maintain MMOCR in the future.

22.1.2 New Features & Enhancements

- Convert numpy.float32 type to python built-in float type by @JunYao1020 in https://github.com/open-mmlab/mmocr/pull/1462
- When '.' char not in output string, output is also considered to be a … by @JunYao1020 in https://github.com/open-mmlab/mmocr/pull/1457
- Refactor issue template by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1449
- issue template by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1489
- Update maintainers by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1504
- Support MMCV < 1.8.0 by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1508

22.1.3 Bug Fixes

- fix ci by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1491
- [CI] Fix CI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1463

22.1.4 Docs

- [DOCs] Add MMYOLO in Readme. by @ysh329 in https://github.com/open-mmlab/mmocr/pull/1475
- [Docs] Update contributing.md by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1490

22.1.5 New Contributors

- @ysh329 made their first contribution in https://github.com/open-mmlab/mmocr/pull/1475

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v0.6.2···v0.6.3

22.2 0.6.2 (14/10/2022)

22.2.1 Highlights

It’ s now possible to train/test models through Python Interface. For example, you can train a model under mmocr/ directory in this way:

```
# an example of how to use such modifications is shown as the following:
from mmocr.tools.train import TrainArg, parse_args, run_train_cmd
args = TrainArg(config='/path/to/config.py')
args.add_arg('--work-dir', '/path/to/dir')
args = parse_args(args.arg_list)
run_train_cmd(args)
```

See PR #1138 for more details.

Besides, release candidates for MMOCR 1.0 with tons of new features are available at 1.x branch now! Check out the changelog for more information about the features, and maintenance plan for how we will maintain MMOCR in the future.

22.2.2 New Features

- Adding test & train API to be used directly in code by @wybryan in https://github.com/open-mmlab/mmocr/pull/1138
- Let ResizeOCR full support mmcv.impad's pad_val parameters by @hsiehpinghan in https://github.com/open-mmlab/mmocr/pull/1437

22.2.3 Bug Fixes

- Fix ABINet config by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1256
- Fix Recognition Score Normalization Issue by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1333
- Remove max_seq_len inconsistency by @antoniolanza1996 in https://github.com/open-mmlab/mmocr/pull/1433
- box points ordering by @yjmm10 in https://github.com/open-mmlab/mmocr/pull/1205
- Correct spelling by misspelling 'preperties' to 'properties' by @JunYao1020 in https://github.com/open-mmlab/mmocr/pull/1446

22.2.4 Docs

- Demo, experiments and live inference API on Tiyaro by @Venkat2811 in https://github.com/open-mmlab/mmocr/pull/1272
- Update 1.x info by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1369
- Add global notes to the docs and the version switcher menu by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1406
- Logger Hook Config Updated to Add WandB by @Nourollah in https://github.com/open-mmlab/mmocr/pull/1345

22.2.5 New Contributors

- @Venkat2811 made their first contribution in https://github.com/open-mmlab/mmocr/pull/1272
- @wybryan made their first contribution in https://github.com/open-mmlab/mmocr/pull/1139
- @hsiehpinghan made their first contribution in https://github.com/open-mmlab/mmocr/pull/1437
- @yjmm10 made their first contribution in https://github.com/open-mmlab/mmocr/pull/1205
- @JunYao1020 made their first contribution in https://github.com/open-mmlab/mmocr/pull/1446
- @Nourollah made their first contribution in https://github.com/open-mmlab/mmocr/pull/1345

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v0.6.1…v0.6.2

22.3 0.6.1 (04/08/2022)

22.3.1 Highlights

1. ArT dataset is available for text detection and recognition!
2. Fix several bugs that affects the correctness of the models.
3. Thanks to MIM, our installation is much simpler now! The docs has been renewed as well.

22.3.2 New Features & Enhancements

- Add ArT by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1006
- add ABINet_Vision api by @Abdelrahman350 in https://github.com/open-mmlab/mmocr/pull/1041
- add codespell ignore and use mdformat by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/1022
- Add mim to extras_requrie to setup.py, update mminstall…by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1062
- Simplify normalized edit distance calculation by @maxbachmann in https://github.com/open-mmlab/mmocr/pull/1060
- Test mim in CI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1090
- Remove redundant steps by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1091
- Update links to SDMGR links by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1252

22.3.3 Bug Fixes

- Remove unnecessary requirements by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1000
- Remove confusing img_scales in pipelines by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1007
- inplace operator "+="will cause RuntimeError when model backward by @garvan2021 in https://github.com/open-mmlab/mmocr/pull/1018
- Fix a typo problem in MASTER by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1031
- Fix config name of MASTER in ocr.py by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1044
- Relax OpenCV requirement by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1061
- Restrict the minimum version of OpenCV to avoid potential vulnerability by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1065
- typo by @tpoisonooo in https://github.com/open-mmlab/mmocr/pull/1024
- Fix a typo in setup.py by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1095

- fix #1067: add torchserve DockerFile and fix bugs by @Hegelim in https://github.com/open-mmlab/mmocr/pull/1073
- Incorrect filename in labelme_converter.py by @xiefeifeihu in https://github.com/open-mmlab/mmocr/pull/1103
- Fix dataset configs by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1106
- Fix #1098: normalize text recognition scores by @Hegelim in https://github.com/open-mmlab/mmocr/pull/1119
- Update ST_SA_MJ_train.py by @MingyuLau in https://github.com/open-mmlab/mmocr/pull/1117
- PSENet metafile by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1121
- Flexible ways of getting file name by @balandongiv in https://github.com/open-mmlab/mmocr/pull/1107
- Updating edge-embeddings after each GNN layer by @amitbcp in https://github.com/open-mmlab/mmocr/pull/1134
- links update by @TekayaNidham in https://github.com/open-mmlab/mmocr/pull/1141
- bug fix: access params by cfg.get by @doem97 in https://github.com/open-mmlab/mmocr/pull/1145
- Fix a bug in LmdbAnnFileBackend that cause breaking in Synthtext detection training by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1159
- Fix typo of –lmdb-map-size default value by @easilylazy in https://github.com/open-mmlab/mmocr/pull/1147
- Fixed docstring syntax error of line 19 & 21 by @APX103 in https://github.com/open-mmlab/mmocr/pull/1157
- Update lmdb_converter and ct80 cropped image source in document by @doem97 in https://github.com/open-mmlab/mmocr/pull/1164
- MMCV compatibility due to outdated MMDet by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1192
- Update maximum version of mmcv by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1219
- Update ABINet links for main by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1221
- Update owners by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1248
- Add back some missing fields in configs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1171

22.3.4 Docs

- Fix typos by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/1001
- Configure Myst-parser to parse anchor tag by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1012
- Fix a error in docs/en/tutorials/dataset_types.md by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1034
- Update readme according to the guideline by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1047
- Limit markdown version by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1172

- Limit extension versions by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/1210
- Update installation guide by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1254
- Update image link @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/1255

22.3.5 New Contributors

- @tpoisonooo made their first contribution in https://github.com/open-mmlab/mmocr/pull/1024
- @Abdelrahman350 made their first contribution in https://github.com/open-mmlab/mmocr/pull/1041
- @Hegelim made their first contribution in https://github.com/open-mmlab/mmocr/pull/1073
- @xiefeifeihu made their first contribution in https://github.com/open-mmlab/mmocr/pull/1103
- @MingyuLau made their first contribution in https://github.com/open-mmlab/mmocr/pull/1117
- @balandongiv made their first contribution in https://github.com/open-mmlab/mmocr/pull/1107
- @amitbcp made their first contribution in https://github.com/open-mmlab/mmocr/pull/1134
- @TekayaNidham made their first contribution in https://github.com/open-mmlab/mmocr/pull/1141
- @easilylazy made their first contribution in https://github.com/open-mmlab/mmocr/pull/1147
- @APX103 made their first contribution in https://github.com/open-mmlab/mmocr/pull/1157

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v0.6.0…v0.6.1

22.4 0.6.0 (05/05/2022)

22.4.1 Highlights

1. A new recognition algorithm MASTER has been added into MMOCR, which was the championship solution for the "ICDAR 2021 Competition on Scientific Table Image Recognition to Latex"! The model pre-trained on SynthText and MJSynth is available for testing! Credit to @JiaquanYe
2. DBNet++ has been released now! A new Adaptive Scale Fusion module has been equipped for feature enhancement. Benefiting from this, the new model achieved 2% better h-mean score than its predecessor on the ICDAR2015 dataset.
3. Three more dataset converters are added: LSVT, RCTW and HierText. Check the dataset zoo (Det & Recog) to explore further information.
4. To enhance the data storage efficiency, MMOCR now supports loading both images and labels from .lmdb format annotations for the text recognition task. To enable such a feature, the new lmdb_converter.py is ready for use to pack your cropped images and labels into an lmdb file. For a detailed tutorial, please refer to the following sections and the doc.

5. Testing models on multiple datasets is a widely used evaluation strategy. MMOCR now supports automatically reporting mean scores when there is more than one dataset to evaluate, which enables a more convenient comparison between checkpoints. Doc
6. Evaluation is more flexible and customizable now. For text detection tasks, you can set the score threshold range where the best results might come out. (Doc) If too many results are flooding your text recognition train log, you can trim it by specifying a subset of metrics in evaluation config. Check out the Evaluation section for details.
7. MMOCR provides a script to convert the .json labels obtained by the popular annotation toolkit **Labelme** to MMOCR-supported data format. @Y-M-Y contributed a log analysis tool that helps users gain a better understanding of the entire training process. Read tutorial docs to get started.

22.4.2 Lmdb Dataset

Reading images or labels from files can be slow when data are excessive, e.g. on a scale of millions. Besides, in academia, most of the scene text recognition datasets are stored in lmdb format, including images and labels. To get closer to the mainstream practice and enhance the data storage efficiency, MMOCR now officially supports loading images and labels from lmdb datasets via a new pipeline LoadImageFromLMDB. This section is intended to serve as a quick walkthrough for you to master this update and apply it to facilitate your research.

Specifications

To better align with the academic community, MMOCR now requires the following specifications for lmdb datasets:

- The parameter describing the data volume of the dataset is `num-samples` instead of `total_number` (deprecated).
- Images and labels are stored with keys in the form of `image-000000001` and `label-000000001`, respectively.

Usage

1. Use existing academic lmdb datasets if they meet the specifications; or the tool provided by MMOCR to pack images & annotations into a lmdb dataset.

- Previously, MMOCR had a function `txt2lmdb` (deprecated) that only supported converting labels to lmdb format. However, it is quite different from academic lmdb datasets, which usually contain both images and labels. Now MMOCR provides a new utility lmdb_converter to convert recognition datasets with both images and labels to lmdb format.
- Say that your recognition data in MMOCR's format are organized as follows. (See an example in ocr_toy_dataset).

```
# Directory structure

├──img_path
```

(下页继续)

(续上页)

```
|       |—— img1.jpg
|       |—— img2.jpg
|       |—— ...
|——label.txt (or label.jsonl)

# Annotation format

label.txt:  img1.jpg HELLO
            img2.jpg WORLD
            ...

label.jsonl:    {'filename':'img1.jpg', 'text':'HELLO'}
                {'filename':'img2.jpg', 'text':'WORLD'}
                ...
```

- Then pack these files up:

```
python tools/data/utils/lmdb_converter.py  {PATH_TO_LABEL} {OUTPUT_PATH} --i
↪{PATH_TO_IMAGES}
```

- Check out tools.md for more details.

2. The second step is to modify the configuration files. For example, to train CRNN on MJ and ST datasets:

- Set parser as `LineJsonParser` and `file_format` as ‘lmdb’ in dataset config

```
# configs/_base_/recog_datasets/ST_MJ_train.py
train1 = dict(
    type='OCRDataset',
    img_prefix=train_img_prefix1,
    ann_file=train_ann_file1,
    loader=dict(
        type='AnnFileLoader',
        repeat=1,
        file_format='lmdb',
        parser=dict(
            type='LineJsonParser',
            keys=['filename', 'text'],
        )),
    pipeline=None,
    test_mode=False)
```

- Use `LoadImageFromLMDB` in pipeline:

```
# configs/_base_/recog_pipelines/crnn_pipeline.py
train_pipeline = [
    dict(type='LoadImageFromLMDB', color_type='grayscale'),
    ...
```

3. You are good to go! Start training and MMOCR will load data from your lmdb dataset.

22.4.3 New Features & Enhancements

- Add analyze_logs in tools and its description in docs by @Y-M-Y in https://github.com/open-mmlab/mmocr/pull/899
- Add LSVT Data Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/896
- Add RCTW dataset converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/914
- Support computing mean scores in UniformConcatDataset by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/981
- Support loading images and labels from lmdb file by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/982
- Add recog2lmdb and new toy dataset files by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/979
- Add labelme converter for textdet and textrecog by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/972
- Update CircleCI configs by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/918
- Update Git Action by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/930
- More customizable fields in dataloaders by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/933
- Skip CIs when docs are modified by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/941
- Rename Github tests, fix ignored paths by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/946
- Support latest MMCV by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/959
- Support dynamic threshold range in eval_hmean by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/962
- Update the version requirement of mmdet in docker by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/966
- Replace `opencv-python-headless` with `open-python` by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/970
- Update Dataset Configs by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/980
- Add SynthText dataset config by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/983

- Automatically report mean scores when applicable by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/995
- Add DBNet++ by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/973
- Add MASTER by @JiaquanYe in https://github.com/open-mmlab/mmocr/pull/807
- Allow choosing metrics to report in text recognition tasks by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/989
- Add HierText converter by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/948
- Fix lint_only in CircleCI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/998

22.4.4 Bug Fixes

- Fix CircleCi Main Branch Accidentally Run PR Stage Test by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/927
- Fix a deprecate warning about mmdet.datasets.pipelines.formating by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/944
- Fix a Bug in ResNet plugin by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/967
- revert a wrong setting in db_r18 cfg by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/978
- Fix TotalText Anno version issue by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/945
- Update installation step of `albumentations` by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/984
- Fix ImgAug transform by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/949
- Fix GPG key error in CI and docker by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/988
- update label.lmdb by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/991
- correct meta key by @garvan2021 in https://github.com/open-mmlab/mmocr/pull/926
- Use new image by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/976
- Fix Data Converter Issues by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/955

22.4.5 Docs

- Update CONTRIBUTING.md by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/905
- Fix the misleading description in test.py by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/908
- Update recog.md for lmdb Generation by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/934
- Add MMCV by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/954

- Add wechat QR code to CN readme by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/960
- Update CONTRIBUTING.md by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/947
- Use QR codes from MMCV by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/971
- Renew dataset_types.md by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/997

22.4.6 New Contributors

- @Y-M-Y made their first contribution in https://github.com/open-mmlab/mmocr/pull/899

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v0.5.0···v0.6.0

22.5 0.5.0 (31/03/2022)

22.5.1 Highlights

1. MMOCR now supports SPACE recognition! (What a prominent feature!) Users only need to convert the recognition annotations that contain spaces from a plain `.txt` file to JSON line format `.jsonl`, and then revise a few configurations to enable the `LineJsonParser`. For more information, please read our step-by-step tutorial.
2. Tesseract is now available in MMOCR! While MMOCR is more flexible to support various downstream tasks, users might sometimes not be satisfied with DL models and would like to turn to effective legacy solutions. Therefore, we offer this option in `mmocr.utils.ocr` by wrapping Tesseract as a detector and/or recognizer. Users can easily create an MMOCR object by `MMOCR(det=’ Tesseract’ , recog=’ Tesseract’ )`. Credit to @garvan2021
3. We release data converters for **16** widely used OCR datasets, including multiple scenarios such as document, handwritten, and scene text. Now it is more convenient to generate annotation files for these datasets. Check the dataset zoo (Det & Recog) to explore further information.
4. Special thanks to @EighteenSprings @BeyondYourself @yangrisheng, who had actively participated in documentation translation!

22.5.2 Migration Guide - ResNet

Some refactoring processes are still going on. For text recognition models, we unified the `ResNet-like` architectures which are used as backbones. By introducing stage-wise and block-wise plugins, the refactored ResNet is highly flexible to support existing models, like ResNet31 and ResNet45, and other future designs of ResNet variants.

Plugin

- `Plugin` is a module category inherited from MMCV’ s implementation of `PLUGIN_LAYERS`, which can be inserted between each stage of ResNet or into a basicblock. You can find a simple implementation of plugin at mmocr/models/textrecog/plugins/common.py, or click the button below.

```
@PLUGIN_LAYERS.register_module()
class Maxpool2d(nn.Module):
    """A wrapper around nn.Maxpool2d().

    Args:
        kernel_size (int or tuple(int)): Kernel size for max pooling layer
        stride (int or tuple(int)): Stride for max pooling layer
        padding (int or tuple(int)): Padding for pooling layer
    """

    def __init__(self, kernel_size, stride, padding=0, **kwargs):
        super(Maxpool2d, self).__init__()
        self.model = nn.MaxPool2d(kernel_size, stride, padding)

    def forward(self, x):
        """
        Args:
            x (Tensor): Input feature map

        Returns:
            Tensor: The tensor after Maxpooling layer.
        """
        return self.model(x)
```

Stage-wise Plugins

- ResNet is composed of stages, and each stage is composed of blocks. E.g., ResNet18 is composed of 4 stages, and each stage is composed of basicblocks. For each stage, we provide two ports to insert stage-wise plugins by giving `plugins` parameters in ResNet.

```
[port1: before stage] ---> [stage] ---> [port2: after stage]
```

- E.g. Using a ResNet with four stages as example. Suppose we want to insert an additional convolution layer before each stage, and an additional convolution layer at stage 1, 2, 4. Then you can define the special ResNet18 like this

```
resnet18_speical = ResNet(
        # for simplicity, some required
        # parameters are omitted
```

(下页继续)

(续上页)

```
        plugins=[
            dict(
                cfg=dict(
                type='ConvModule',
                kernel_size=3,
                stride=1,
                padding=1,
                norm_cfg=dict(type='BN'),
                act_cfg=dict(type='ReLU')),
                stages=(True, True, True, True),
                position='before_stage')
            dict(
                cfg=dict(
                type='ConvModule',
                kernel_size=3,
                stride=1,
                padding=1,
                norm_cfg=dict(type='BN'),
                act_cfg=dict(type='ReLU')),
                stages=(True, True, False, True),
                position='after_stage')
        ])
```

- You can also insert more than one plugin in each port and those plugins will be executed in order. Let' s take ResNet in MASTER as an example:
  - ResNet in Master is based on ResNet31. And after each stage, a module named `GCAModule` will be used. The `GCAModule` is inserted before the stage-wise convolution layer in ResNet31. In conlusion, there will be two plugins at `after_stage` port in the same time.

```
resnet_master = ResNet(
                # for simplicity, some required
                # parameters are omitted
                plugins=[
                    dict(
                        cfg=dict(type='Maxpool2d', kernel_size=2, stride=(2,␣
↪2)),
                        stages=(True, True, False, False),
                        position='before_stage'),
                    dict(
                        cfg=dict(type='Maxpool2d', kernel_size=(2, 1),␣
↪stride=(2, 1)),
                        stages=(False, False, True, False),
                        position='before_stage'),
```

(下页继续)

(续上页)

```
                    dict(
                        cfg=dict(type='GCAModule', kernel_size=3, stride=1,␣
↪padding=1),
                        stages=[True, True, True, True],
                        position='after_stage'),
                    dict(
                        cfg=dict(
                            type='ConvModule',
                            kernel_size=3,
                            stride=1,
                            padding=1,
                            norm_cfg=dict(type='BN'),
                            act_cfg=dict(type='ReLU')),
                        stages=(True, True, True, True),
                        position='after_stage')
                ])
```

– In each plugin, we will pass two parameters (`in_channels`, `out_channels`) to support operations that need the information of current channels.

Block-wise Plugin (Experimental)

- We also refactored the `BasicBlock` used in ResNet. Now it can be customized with block-wise plugins. Check here for more details.
- BasicBlock is composed of two convolution layer in the main branch and a shortcut branch. We provide four ports to insert plugins.

```
    [port1: before_conv1] ---> [conv1] --->
    [port2: after_conv1] ---> [conv2] --->
    [port3: after_conv2] ---> +(shortcut) ---> [port4: after_shortcut]
```

- In each plugin, we will pass a parameter `in_channels` to support operations that need the information of current channels.
- E.g. Build a ResNet with customized BasicBlock with an additional convolution layer before conv1:

```
resnet_31 = ResNet(
        in_channels=3,
        stem_channels=[64, 128],
        block_cfgs=dict(type='BasicBlock'),
        arch_layers=[1, 2, 5, 3],
        arch_channels=[256, 256, 512, 512],
```

(下页继续)

(续上页)

```
        strides=[1, 1, 1, 1],
        plugins=[
            dict(
                cfg=dict(type='Maxpool2d',
                kernel_size=2,
                stride=(2, 2)),
                stages=(True, True, False, False),
                position='before_stage'),
            dict(
                cfg=dict(type='Maxpool2d',
                kernel_size=(2, 1),
                stride=(2, 1)),
                stages=(False, False, True, False),
                position='before_stage'),
            dict(
                cfg=dict(
                type='ConvModule',
                kernel_size=3,
                stride=1,
                padding=1,
                norm_cfg=dict(type='BN'),
                act_cfg=dict(type='ReLU')),
                stages=(True, True, True, True),
                position='after_stage')
        ])
```

Full Examples

- ResNet45 is used in ASTER and ABINet without any plugins.

```
resnet45_aster = ResNet(
    in_channels=3,
    stem_channels=[64, 128],
    block_cfgs=dict(type='BasicBlock', use_conv1x1='True'),
    arch_layers=[3, 4, 6, 6, 3],
    arch_channels=[32, 64, 128, 256, 512],
    strides=[(2, 2), (2, 2), (2, 1), (2, 1), (2, 1)])

resnet45_abi = ResNet(
    in_channels=3,
    stem_channels=32,
    block_cfgs=dict(type='BasicBlock', use_conv1x1='True'),
    arch_layers=[3, 4, 6, 6, 3],
```

(下页继续)

(续上页)

```
    arch_channels=[32, 64, 128, 256, 512],
    strides=[2, 1, 2, 1, 1])
```

- ResNet31 is a typical architecture to use stage-wise plugins. Before the first three stages, Maxpooling layer is used. After each stage, a convolution layer with BN and ReLU is used.

```
resnet_31 = ResNet(
    in_channels=3,
    stem_channels=[64, 128],
    block_cfgs=dict(type='BasicBlock'),
    arch_layers=[1, 2, 5, 3],
    arch_channels=[256, 256, 512, 512],
    strides=[1, 1, 1, 1],
    plugins=[
        dict(
            cfg=dict(type='Maxpool2d',
            kernel_size=2,
            stride=(2, 2)),
            stages=(True, True, False, False),
            position='before_stage'),
        dict(
            cfg=dict(type='Maxpool2d',
            kernel_size=(2, 1),
            stride=(2, 1)),
            stages=(False, False, True, False),
            position='before_stage'),
        dict(
            cfg=dict(
            type='ConvModule',
            kernel_size=3,
            stride=1,
            padding=1,
            norm_cfg=dict(type='BN'),
            act_cfg=dict(type='ReLU')),
            stages=(True, True, True, True),
            position='after_stage')
    ])
```

22.5.3 Migration Guide - Dataset Annotation Loader

The annotation loaders, `LmdbLoader` and `HardDiskLoader`, are unified into `AnnFileLoader` for a more consistent design and wider support on different file formats and storage backends. `AnnFileLoader` can load the annotations from `disk`(default), `http` and `petrel` backend, and parse the annotation in `txt` or `lmdb` format. `LmdbLoader` and `HardDiskLoader` are deprecated, and users are recommended to modify their configs to use the new `AnnFileLoader`. Users can migrate their legacy loader `HardDiskLoader` referring to the following example:

```
# Legacy config
train = dict(
    type='OCRDataset',
    ...
    loader=dict(
        type='HardDiskLoader',
        ...))

# Suggested config
train = dict(
    type='OCRDataset',
    ...
    loader=dict(
        type='AnnFileLoader',
        file_storage_backend='disk',
        file_format='txt',
        ...))
```

Similarly, using `AnnFileLoader` with `file_format='lmdb'` instead of `LmdbLoader` is strongly recommended.

22.5.4 New Features & Enhancements

- Update mmcv install by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/775
- Upgrade isort by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/771
- Automatically infer device for inference if not speicifed by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/781
- Add open-mmlab precommit hooks by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/787
- Add windows CI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/790
- Add CurvedSyntext150k Converter by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/719
- Add FUNSD Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/808

- Support loading annotation file with petrel/http backend by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/793
- Support different seeds on different ranks by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/820
- Support json in recognition converter by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/844
- Add args and docs for multi-machine training/testing by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/849
- Add warning info for LineStrParser by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/850
- Deploy openmmlab-bot by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/876
- Add Tesserocr Inference by @garvan2021 in https://github.com/open-mmlab/mmocr/pull/814
- Add LV Dataset Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/871
- Add SROIE Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/810
- Add NAF Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/815
- Add DeText Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/818
- Add IMGUR Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/825
- Add ILST Converter by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/833
- Add KAIST Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/835
- Add IC11 (Born-digital Images) Data Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/857
- Add IC13 (Focused Scene Text) Data Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/861
- Add BID Converter by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/862
- Add Vintext Converter by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/864
- Add MTWI Data Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/867
- Add COCO Text v2 Data Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/872
- Add ReCTS Data Converter by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/892
- Refactor ResNets by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/809

22.5.5 Bug Fixes

- Bump mmdet version to 2.20.0 in Dockerfile by @GPhilo in https://github.com/open-mmlab/mmocr/pull/763
- Update mmdet version limit by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/773
- Minimum version requirement of albumentations by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/769
- Disable worker in the dataloader of gpu unit test by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/780
- Standardize the type of torch.device in ocr.py by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/800
- Use RECOGNIZER instead of DETECTORS by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/685
- Add num_classes to configs of ABINet by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/805
- Support loading space character from dict file by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/854
- Description in tools/data/utils/txt2lmdb.py by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/870
- ignore_index in SARLoss by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/869
- Fix a bug that may cause inplace operation error by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/884
- Use hyphen instead of underscores in script args by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/890

22.5.6 Docs

- Add deprecation message for deploy tools by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/801
- Reorganizing OpenMMLab projects in readme by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/806
- Add demo/README_zh.md by @EighteenSprings in https://github.com/open-mmlab/mmocr/pull/802
- Add detailed version requirement table by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/778
- Correct misleading section title in training.md by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/819
- Update README_zh-CN document URL by @BeyondYourself in https://github.com/open-mmlab/mmocr/pull/823
- translate testing.md. by @yangrisheng in https://github.com/open-mmlab/mmocr/pull/822

- Fix confused description for load-from and resume-from by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/842
- Add documents getting_started in docs/zh by @BeyondYourself in https://github.com/open-mmlab/mmocr/pull/841
- Add the model serving translation document by @BeyondYourself in https://github.com/open-mmlab/mmocr/pull/845
- Update docs about installation on Windows by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/852
- Update tutorial notebook by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/853
- Update Instructions for New Data Converters by @xinke-wang in https://github.com/open-mmlab/mmocr/pull/900
- Brief installation instruction in README by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/897
- update doc for ILST, VinText, BID by @Mountchicken in https://github.com/open-mmlab/mmocr/pull/902
- Fix typos in readme by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/903
- Recog dataset doc by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/893
- Reorganize the directory structure section in det.md by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/894

22.6 New Contributors

- @GPhilo made their first contribution in https://github.com/open-mmlab/mmocr/pull/763
- @xinke-wang made their first contribution in https://github.com/open-mmlab/mmocr/pull/801
- @EighteenSprings made their first contribution in https://github.com/open-mmlab/mmocr/pull/802
- @BeyondYourself made their first contribution in https://github.com/open-mmlab/mmocr/pull/823
- @yangrisheng made their first contribution in https://github.com/open-mmlab/mmocr/pull/822
- @Mountchicken made their first contribution in https://github.com/open-mmlab/mmocr/pull/844
- @garvan2021 made their first contribution in https://github.com/open-mmlab/mmocr/pull/814

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v0.4.1…v0.5.0

22.7 v0.4.1 (27/01/2022)

22.7.1 Highlights

1. Visualizing edge weights in OpenSet KIE is now supported! https://github.com/open-mmlab/mmocr/pull/677
2. Some configurations have been optimized to significantly speed up the training and testing processes! Don't worry - you can still tune these parameters in case these modifications do not work. https://github.com/open-mmlab/mmocr/pull/757
3. Now you can use CPU to train/debug your model! https://github.com/open-mmlab/mmocr/pull/752
4. We have fixed a severe bug that causes users unable to call `mmocr.apis.test` with our pre-built wheels. https://github.com/open-mmlab/mmocr/pull/667

22.7.2 New Features & Enhancements

- Show edge score for openset kie by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/677
- Download flake8 from github as pre-commit hooks by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/695
- Deprecate the support for 'python setup.py test' by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/722
- Disable multi-processing feature of cv2 to speed up data loading by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/721
- Extend ctw1500 converter to support text fields by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/729
- Extend totaltext converter to support text fields by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/728
- Speed up training by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/739
- Add setup multi-processing both in train and test.py by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/757
- Support CPU training/testing by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/752
- Support specify gpu for testing and training with gpu-id instead of gpu-ids and gpus by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/756
- Remove unnecessary custom_import from test.py by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/758

22.7.3 Bug Fixes

- Fix satrn onnxruntime test by @AllentDan in https://github.com/open-mmlab/mmocr/pull/679
- Support both ConcatDataset and UniformConcatDataset by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/675
- Fix bugs of show_results in single_gpu_test by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/667
- Fix a bug for sar decoder when bi-rnn is used by @MhLiao in https://github.com/open-mmlab/mmocr/pull/690
- Fix opencv version to avoid some bugs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/694
- Fix py39 ci error by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/707
- Update visualize.py by @TommyZihao in https://github.com/open-mmlab/mmocr/pull/715
- Fix link of config by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/726
- Use yaml.safe_load instead of load by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/753
- Add necessary keys to test_pipelines to enable test-time visualization by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/754

22.7.4 Docs

- Fix recog.md by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/674
- Add config tutorial by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/683
- Add MMSelfSup/MMRazor/MMDeploy in readme by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/692
- Add recog & det model summary by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/693
- Update docs link by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/710
- add pull request template.md by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/711
- Add website links to readme by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/731
- update readme according to standard by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/742

22.7.5 New Contributors

- @MhLiao made their first contribution in https://github.com/open-mmlab/mmocr/pull/690
- @TommyZihao made their first contribution in https://github.com/open-mmlab/mmocr/pull/715

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v0.4.0…v0.4.1

22.8 v0.4.0 (15/12/2021)

22.8.1 Highlights

1. We release a new text recognition model - ABINet (CVPR 2021, Oral). With it dedicated model design and useful data augmentation transforms, ABINet can achieve the best performance on irregular text recognition tasks. Check it out!
2. We are also working hard to fulfill the requests from our community. OpenSet KIE is one of the achievement, which extends the application of SDMGR from text node classification to node-pair relation extraction. We also provide a demo script to convert WildReceipt to open set domain, though it cannot take the full advantage of OpenSet format. For more information, please read our tutorial.
3. APIs of models can be exposed through TorchServe. Docs

22.8.2 Breaking Changes & Migration Guide

Postprocessor

Some refactoring processes are still going on. For all text detection models, we unified their `decode` implementations into a new module category, `POSTPROCESSOR`, which is responsible for decoding different raw outputs into boundary instances. In all text detection configs, the `text_repr_type` argument in `bbox_head` is deprecated and will be removed in the future release.

Migration Guide: Find a similar line from detection model’s config:

```
text_repr_type=xxx,
```

And replace it with

```
postprocessor=dict(type='{MODEL_NAME}Postprocessor', text_repr_type=xxx)),
```

Take a snippet of PANet’s config as an example. Before the change, its config for `bbox_head` looks like:

```
    bbox_head=dict(
        type='PANHead',
        text_repr_type='poly',
        in_channels=[128, 128, 128, 128],
        out_channels=6,
        loss=dict(type='PANLoss')),
```

Afterwards:

```
    bbox_head=dict(
    type='PANHead',
    in_channels=[128, 128, 128, 128],
    out_channels=6,
    loss=dict(type='PANLoss'),
    postprocessor=dict(type='PANPostprocessor', text_repr_type='poly')),
```

There are other postprocessors and each takes different arguments. Interested users can find their interfaces or implementations in `mmocr/models/textdet/postprocess` or through our api docs.

New Config Structure

We reorganized the `configs/` directory by extracting reusable sections into `configs/_base_`. Now the directory tree of `configs/_base_` is organized as follows:

```
_base_
├── det_datasets
├── det_models
├── det_pipelines
├── recog_datasets
├── recog_models
├── recog_pipelines
└── schedules
```

Most of model configs are making full use of base configs now, which makes the overall structural clearer and facilitates fair comparison across models. Despite the seemingly significant hierarchical difference, **these changes would not break the backward compatibility** as the names of model configs remain the same.

22.8.3 New Features

- Support openset kie by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/498
- Add converter for the Open Images v5 text annotations by Krylov et al. by @baudm in https://github.com/open-mmlab/mmocr/pull/497
- Support Chinese for kie show result by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/464
- Add TorchServe support for text detection and recognition by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/522
- Save filename in text detection test results by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/570
- Add codespell pre-commit hook and fix typos by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/520
- Avoid duplicate placeholder docs in CN by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/582

- Save results to json file for kie. by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/589
- Add SAR_CN to ocr.py by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/579
- mim extension for windows by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/641
- Support muitiple pipelines for different datasets by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/657
- ABINet Framework by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/651

22.8.4 Refactoring

- Refactor textrecog config structure by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/617
- Refactor text detection config by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/626
- refactor transformer modules by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/618
- refactor textdet postprocess by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/640

22.8.5 Docs

- C++ example section by @apiaccess21 in https://github.com/open-mmlab/mmocr/pull/593
- install.md Chinese section by @A465539338 in https://github.com/open-mmlab/mmocr/pull/364
- Add Chinese Translation of deployment.md. by @fatfishZhao in https://github.com/open-mmlab/mmocr/pull/506
- Fix a model link and add the metafile for SATRN by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/473
- Improve docs style by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/474
- Enhancement & sync Chinese docs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/492
- TorchServe docs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/539
- Update docs menu by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/564
- Docs for KIE CloseSet & OpenSet by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/573
- Fix broken links by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/576
- Docstring for text recognition models by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/562
- Add MMFlow & MIM by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/597
- Add MMFewShot by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/621
- Update model readme by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/604
- Add input size check to model_inference by @mpena-vina in https://github.com/open-mmlab/mmocr/pull/633
- Docstring for textdet models by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/561

- Add MMHuman3D in readme by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/644
- Use shared menu from theme instead by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/655
- Refactor docs structure by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/662
- Docs fix by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/664

22.8.6 Enhancements

- Use bounding box around polygon instead of within polygon by @alexander-soare in https://github.com/open-mmlab/mmocr/pull/469
- Add CITATION.cff by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/476
- Add py3.9 CI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/475
- update model-index.yml by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/484
- Use container in CI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/502
- CircleCI Setup by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/611
- Remove unnecessary custom_import from train.py by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/603
- Change the upper version of mmcv to 1.5.0 by @zhouzaida in https://github.com/open-mmlab/mmocr/pull/628
- Update CircleCI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/631
- Pass custom_hooks to MMCV by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/609
- Skip CI when some specific files were changed by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/642
- Add markdown linter in pre-commit hook by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/643
- Use shape from loaded image by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/652
- Cancel previous runs that are not completed by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/666

22.8.7 Bug Fixes

- Modify algorithm "sar" weights path in metafile by @ShoupingShan in https://github.com/open-mmlab/mmocr/pull/581
- Fix Cuda CI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/472
- Fix image export in test.py for KIE models by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/486
- Allow invalid polygons in intersection and union by default by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/471

- Update checkpoints’ links for SATRN by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/518
- Fix converting to onnx bug because of changing key from img_shape to resize_shape by @Harold-lkk in https://github.com/open-mmlab/mmocr/pull/523
- Fix PyTorch 1.6 incompatible checkpoints by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/540
- Fix paper field in metafiles by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/550
- Unify recognition task names in metafiles by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/548
- Fix py3.9 CI by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/563
- Always map location to cpu when loading checkpoint by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/567
- Fix wrong model builder in recog_test_imgs by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/574
- Improve dbnet r50 by fixing img std by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/578
- Fix resource warning: unclosed file by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/577
- Fix bug that same start_point for different texts in draw_texts_by_pil by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/587
- Keep original texts for kie by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/588
- Fix random seed by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/600
- Fix DBNet_r50 config by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/625
- Change SBC case to DBC case by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/632
- Fix kie demo by @innerlee in https://github.com/open-mmlab/mmocr/pull/610
- fix type check by @cuhk-hbsun in https://github.com/open-mmlab/mmocr/pull/650
- Remove depreciated image validator in totaltext converter by @gaotongxiao in https://github.com/open-mmlab/mmocr/pull/661
- Fix change locals() dict by @Fei-Wang in https://github.com/open-mmlab/mmocr/pull/663
- fix #614: textsnake targets by @HolyCrap96 in https://github.com/open-mmlab/mmocr/pull/660

22.8.8 New Contributors

- @alexander-soare made their first contribution in https://github.com/open-mmlab/mmocr/pull/469
- @A465539338 made their first contribution in https://github.com/open-mmlab/mmocr/pull/364
- @fatfishZhao made their first contribution in https://github.com/open-mmlab/mmocr/pull/506
- @baudm made their first contribution in https://github.com/open-mmlab/mmocr/pull/497
- @ShoupingShan made their first contribution in https://github.com/open-mmlab/mmocr/pull/581

- @apiaccess21 made their first contribution in https://github.com/open-mmlab/mmocr/pull/593
- @zhouzaida made their first contribution in https://github.com/open-mmlab/mmocr/pull/628
- @mpena-vina made their first contribution in https://github.com/open-mmlab/mmocr/pull/633
- @Fei-Wang made their first contribution in https://github.com/open-mmlab/mmocr/pull/663

Full Changelog: https://github.com/open-mmlab/mmocr/compare/v0.3.0…0.4.0

22.9 v0.3.0 (25/8/2021)

22.9.1 Highlights

1. We add a new text recognition model –SATRN! Its pretrained checkpoint achieves the best performance over other provided text recognition models. A lighter version of SATRN is also released which can obtain ~98% of the performance of the original model with only 45 MB in size. (@2793145003) #405
2. Improve the demo script, `ocr.py`, which supports applying end-to-end text detection, text recognition and key information extraction models on images with easy-to-use commands. Users can find its full documentation in the demo section. (@samayala22, @manjrekarom) #371, #386, #400, #374, #428
3. Our documentation is reorganized into a clearer structure. More useful contents are on the way! #409, #454
4. The requirement of `Polygon3` is removed since this project is no longer maintained or distributed. We unified all its references to equivalent substitutions in `shapely` instead. #448

22.9.2 Breaking Changes & Migration Guide

1. Upgrade version requirement of MMDetection to 2.14.0 to avoid bugs #382
2. MMOCR now has its own model and layer registries inherited from MMDetection’s or MMCV’s counterparts. (#436) The modified hierarchical structure of the model registries are now organized as follows.

```
mmcv.MODELS -> mmdet.BACKBONES -> BACKBONES
mmcv.MODELS -> mmdet.NECKS -> NECKS
mmcv.MODELS -> mmdet.ROI_EXTRACTORS -> ROI_EXTRACTORS
mmcv.MODELS -> mmdet.HEADS -> HEADS
mmcv.MODELS -> mmdet.LOSSES -> LOSSES
mmcv.MODELS -> mmdet.DETECTORS -> DETECTORS
mmcv.ACTIVATION_LAYERS -> ACTIVATION_LAYERS
mmcv.UPSAMPLE_LAYERS -> UPSAMPLE_LAYERS
```

To migrate your old implementation to our new backend, you need to change the import path of any registries and their corresponding builder functions (including `build_detectors`) from `mmdet.models.builder` to `mmocr.`

`models.builder`. If you have referred to any model or layer of MMDetection or MMCV in your model config, you need to add `mmdet.` or `mmcv.` prefix to its name to inform the model builder of the right namespace to work on.

Interested users may check out MMCV’ s tutorial on Registry for in-depth explanations on its mechanism.

22.9.3 New Features

- Automatically replace SyncBN with BN for inference #420, #453
- Support batch inference for CRNN and SegOCR #407
- Support exporting documentation in pdf or epub format #406
- Support `persistent_workers` option in data loader #459

22.9.4 Bug Fixes

- Remove depreciated key in kie_test_imgs.py #381
- Fix dimension mismatch in batch testing/inference of DBNet #383
- Fix the problem of dice loss which stays at 1 with an empty target given #408
- Fix a wrong link in ocr.py (@naarkhoo) #417
- Fix undesired assignment to “pretrained” in test.py #418
- Fix a problem in polygon generation of DBNet #421, #443
- Skip invalid annotations in totaltext_converter #438
- Add zero division handler in poly utils, remove Polygon3 #448

22.9.5 Improvements

- Replace lanms-proper with lanms-neo to support installation on Windows (with special thanks to @gen-ko who has re-distributed this package!)
- Support MIM #394
- Add tests for PyTorch 1.9 in CI #401
- Enables fullscreen layout in readthedocs #413
- General documentation enhancement #395
- Update version checker #427
- Add copyright info #439
- Update citation information #440

22.9.6 Contributors

We thank @2793145003, @samayala22, @manjrekarom, @naarkhoo, @gen-ko, @duanjiaqi, @gaotongxiao, @cuhk-hbsun, @innerlee, @wdsd641417025 for their contribution to this release!

22.10 v0.2.1 (20/7/2021)

22.10.1 Highlights

1. Upgrade to use MMCV-full **>= 1.3.8** and MMDetection **>= 2.13.0** for latest features
2. Add ONNX and TensorRT export tool, supporting the deployment of DBNet, PSENet, PANet and CRNN (experimental) #278, #291, #300, #328
3. Unified parameter initialization method which uses init_cfg in config files #365

22.10.2 New Features

- Support TextOCR dataset #293
- Support Total-Text dataset #266, #273, #357
- Support grouping text detection box into lines #290, #304
- Add benchmark_processing script that benchmarks data loading process #261
- Add SynthText preprocessor for text recognition models #351, #361
- Support batch inference during testing #310
- Add user-friendly OCR inference script #366

22.10.3 Bug Fixes

- Fix improper class ignorance in SDMGR Loss #221
- Fix potential numerical zero division error in DRRG #224
- Fix installing requirements with pip and mim #242
- Fix dynamic input error of DBNet #269
- Fix space parsing error in LineStrParser #285
- Fix textsnake decode error #264
- Correct isort setup #288
- Fix a bug in SDMGR config #316

- Fix kie_test_img for KIE nonvisual #319
- Fix metafiles #342
- Fix different device problem in FCENet #334
- Ignore improper tailing empty characters in annotation files #358
- Docs fixes #247, #255, #265, #267, #268, #270, #276, #287, #330, #355, #367
- Fix NRTR config #356, #370

22.10.4 Improvements

- Add backend for resizeocr #244
- Skip image processing pipelines in SDMGR novisual #260
- Speedup DBNet #263
- Update mmcv installation method in workflow #323
- Add part of Chinese documentations #353, #362
- Add support for ConcatDataset with two workflows #348
- Add list_from_file and list_to_file utils #226
- Speed up sort_vertex #239
- Support distributed evaluation of KIE #234
- Add pretrained FCENet on IC15 #258
- Support CPU for OCR demo #227
- Avoid extra image pre-processing steps #375

22.11 v0.2.0 (18/5/2021)

22.11.1 Highlights

1. Add the NER approach Bert-softmax (NAACL’2019)
2. Add the text detection method DRRG (CVPR’2020)
3. Add the text detection method FCENet (CVPR’2021)
4. Increase the ease of use via adding text detection and recognition end-to-end demo, and colab online demo.
5. Simplify the installation.

22.11.2 New Features

- Add Bert-softmax for Ner task #148
- Add DRRG #189
- Add FCENet #133
- Add end-to-end demo #105
- Support batch inference #86 #87 #178
- Add TPS preprocessor for text recognition #117 #135
- Add demo documentation #151 #166 #168 #170 #171
- Add checkpoint for Chinese recognition #156
- Add metafile #175 #176 #177 #182 #183
- Add support for numpy array inference #74

22.11.3 Bug Fixes

- Fix the duplicated point bug due to transform for textsnake #130
- Fix CTC loss NaN #159
- Fix error raised if result is empty in demo #144
- Fix results missing if one image has a large number of boxes #98
- Fix package missing in dockerfile #109

22.11.4 Improvements

- Simplify installation procedure via removing compiling #188
- Speed up panet post processing so that it can detect dense texts #188
- Add zh-CN README #70 #95
- Support windows #89
- Add Colab #147 #199
- Add 1-step installation using conda environment #193 #194 #195

22.12 v0.1.0 (7/4/2021)

22.12.1 Highlights

- MMOCR is released.

22.12.2 Main Features

- Support text detection, text recognition and the corresponding downstream tasks such as key information extraction.
- For text detection, support both single-step (`PSENet`, `PANet`, `DBNet`, `TextSnake`) and two-step (`MaskRCNN`) methods.
- For text recognition, support CTC-loss based method `CRNN`; Encoder-decoder (with attention) based methods `SAR`, `Robustscanner`; Segmentation based method `SegOCR`; Transformer based method `NRTR`.
- For key information extraction, support GCN based method `SDMG-R`.
- Provide checkpoints and log files for all of the methods above.

CHAPTER 23

mmocr.apis

CHAPTER 24

mmocr.core

24.1 evaluation

CHAPTER 25

mmocr.utils

CHAPTER 26

mmocr.models

26.1 Common Backbones

26.2 Text Detection Detectors

26.3 Text Detection Heads

26.4 Text Detection Necks

26.5 Text Detection Losses

26.6 Text Detection Postprocessors

26.7 Text Recognition Recognizer

26.8 Text Recognition Backbones

26.9 Text Recognition Necks

26.10 Text Recognition Heads

26.11 Text Recognition Preprocessors

CHAPTER 27

mmocr.datasets

27.1 datasets

27.2 pipelines

27.3 utils

CHAPTER 28

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 29

导引

- genindex
- search