MMOCR

发布 *0.3.0*

OpenMMLab

2021 年 12 月 10 日

开始

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 暂未获得官方支持)
- 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
- MMCV >= 1.3.8
- MMDetection >= 2.14.0

我们已经测试了以下操作系统和软件版本:

- OS: Ubuntu 16.04
- CUDA: 10.1
- GCC(G++): 5.4.0
- MMCV 1.3.8
- MMDetection 2.14.0

- PyTorch 1.6.0
- torchvision 0.7.0

MMOCR 基于 PyTorch 和 MMDetection 项目实现。

1.2 详细安装步骤

a. 创建一个 Conda 虚拟环境并激活（open-mmlab 为自定义环境名）。

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

b. 按照 PyTorch 官网教程安装 PyTorch 和 torchvision (参见官方链接), 例如,

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

注解： 请确定 CUDA 编译版本和运行版本一致。你可以在 PyTorch 官网检查预编译 PyTorch 所支持的 CUDA 版本。

---

c. 安装 mmcv，推荐以下方式进行安装。

```
pip install mmcv-full -f https://download.openmmlab.com/mmcv/dist/{cu_version}/{torch_
↪version}/index.html
```

请将上述 url 中 `{cu_version}` 和 `{torch_version}` 替换成你环境中对应的 CUDA 版本和 PyTorch 版本。例如，如果想要安装最新版基于 CUDA 11 和 PyTorch 1.7.0 的最新版 `mmcv-full`，请输入以下命令:

```
pip install mmcv-full -f https://download.openmmlab.com/mmcv/dist/cu110/torch1.7.0/
↪index.html
```

注解： 使用 mmocr 0.2.0 及更高版本需要安装 mmcv 1.3.4 或更高版本。

如果安装时进行了编译过程，请再次确认安装的 `mmcv-full` 版本与环境中 CUDA 和 PyTorch 的版本匹配。即使是 PyTorch 1.7.0 和 1.7.1，`mmcv-full` 的安装版本也是有区别的。

如有需要，可以在此处检查 mmcv 与 CUDA 和 PyTorch 的版本对应关系。

警告： 如果你已经安装过 `mmcv`，你需要先运行 `pip uninstall mmcv` 删除 `mmcv`，再安装 `mmcv-full`。如果环境中同时安装了 `mmcv` 和 `mmcv-full`, 将会出现报错 `ModuleNotFoundError`。

d. 安装 mmdet, 我们推荐使用 pip 安装最新版 mmdet。在 此处 可以查看 mmdet 版本信息.

```
pip install mmdet
```

或者，你也可以按照 安装指南 中的方法安装 mmdet。

e. 克隆 MMOCR 项目到本地.

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

f. 安装依赖软件环境并安装 MMOCR。

```
pip install -r requirements.txt
pip install -v -e . # or "python setup.py develop"
export PYTHONPATH=$(pwd):$PYTHONPATH
```

1.3 完整安装命令

以下是 conda 方式安装 mmocr 的完整安装命令。

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

# 安装最新的 PyTorch 预编译包
conda install pytorch==1.6.0 torchvision==0.7.0 cudatoolkit=10.1 -c pytorch

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

# 安装 mmdet
pip install mmdet

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

pip install -r requirements.txt
pip install -v -e .  # 或 "python setup.py develop"
export PYTHONPATH=$(pwd):$PYTHONPATH
```

1.4 可选方式: Docker 镜像

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 数据集准备

我们推荐建立一个 symlink 路径映射，连接数据集路径到 `mmocr/data`。详细数据集准备方法请阅读**数据集**章节。如果你需要的文件夹路径不同，你可能需要在 configs 文件中修改对应的文件路径信息。

`mmocr` 文件夹路径结构如下：

```
├── configs/
├── demo/
├── docker/
├── docs/
├── LICENSE
├── mmocr/
├── README.md
├── requirements/
├── requirements.txt
├── resources/
├── setup.cfg
├── setup.py
├── tests/
├── tools/
```

CHAPTER 2

Getting Started

In this guide we will show you some useful commands and familiarize you with MMOCR. We also provide a notebook that can help you get the most out of MMOCR.

2.1 Installation

Check out our *installation guide* for full steps.

2.2 Dataset Preparation

MMOCR supports numerous datasets which are classified by the type of their corresponding tasks. You may find their preparation steps in these sections: *Detection Datasets*, *Recognition Datasets*, *KIE Datasets* and *NER Datasets*.

2.3 Inference with Pretrained Models

You can perform end-to-end OCR on our demo image with one simple line of command:

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

Its detection result will be printed out and a new window will pop up with result visualization. More demo and full instructions can be found in *Demo*.

2.4 Training

2.4.1 Training with Toy Dataset

We provide a toy dataset under `tests/data` on which you can get a sense of training before the academic dataset is prepared.

For example, to train a text recognition task with `seg` method and toy dataset,

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

To train a text recognition task with `sar` method and toy dataset,

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

2.4.2 Training with Academic Dataset

Once you have prepared required academic dataset following our instruction, the only last thing to check is if the model's config points MMOCR to the correct dataset path. Suppose we want to train DBNet on ICDAR 2015, and part of `configs/textdet/dbnet/dbnet_r18_fpnc_1200e_icdar2015.py` looks like the following:

```
dataset_type = 'IcdarDataset'
data_root = 'data/icdar2015'
data = dict(
    train=dict(
        type=dataset_type,
        ann_file=data_root + '/instances_training.json',
        img_prefix=data_root + '/imgs',
        pipeline=train_pipeline)
    val=dict(
        type=dataset_type,
        ann_file=data_root + '/instances_test.json',
        img_prefix=data_root + '/imgs',
        pipeline=test_pipeline),
    test=dict(
        type=dataset_type,
        ann_file=data_root + '/instances_test.json',
        img_prefix=data_root + '/imgs',
        pipeline=test_pipeline))
```

You would need to check if `data/icdar2015` is right. Then you can start training with the command:

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

You can find full training instructions, explanations and useful training configs in *Training*.

2.5 Testing

Suppose now you have finished the training of DBNet and the latest model has been saved in `dbnet/latest.pth`. You can evaluate its performance on the test set using the `hmean-iou` metric with the following command:

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

Evaluating any pretrained model accessible online is also allowed:

```
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
```

More instructions on testing are available in *Testing*.

CHAPTER 3

Demo

We provide an easy-to-use API for the demo and application purpose in ocr.py script.

The API can be called through command line (CL) or by calling it from another python script.

3.1 Example 1: Text Detection

Instruction: Perform detection inference on an image with the TextSnake recognition model, export the result in a json file (default) and save the visualization file.

- CL interface:

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

- Python interface:

```
from mmocr.utils.ocr import MMOCR

# Load models into memory
ocr = MMOCR(det='TextSnake', recog=None)

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

3.2 Example 2: Text Recognition

Instruction: Perform batched recognition inference on a folder with hundreds of image with the CRNN_TPS recognition model and save the visualization results in another folder. *Batch size is set to 10 to prevent out of memory CUDA runtime errors.*

- CL interface:

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

- Python interface:

```
from mmocr.utils.ocr import MMOCR

# Load models into memory
ocr = MMOCR(det=None, recog='CRNN_TPS')

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

3.3 Example 3: Text Detection + Recognition

Instruction: Perform ocr (det + recog) inference on the demo/demo_text_det.jpg image with the PANet_IC15 (default) detection model and SAR (default) recognition model, print the result in the terminal and show the visualization.

- CL interface:

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

注解: When calling the script from the command line, the script assumes configs are saved in the `configs/` folder. User can customize the directory by specifying the value of `config_dir`.

- Python interface:

```
from mmocr.utils.ocr import MMOCR

# Load models into memory
ocr = MMOCR()
```

(下页继续)

(续上页)

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

3.4 Example 4: Text Detection + Recognition + Key Information Extraction

Instruction: Perform end-to-end ocr (det + recog) inference first with PS_CTW detection model and SAR recognition model, then run KIE inference with SDMGR model on the ocr result and show the visualization.

- CL interface:

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

注解: Note: When calling the script from the command line, the script assumes configs are saved in the `configs/` folder. User can customize the directory by specifying the value of `config_dir`.

- Python interface:

```
from mmocr.utils.ocr import MMOCR

# Load models into memory
ocr = MMOCR(det='PS_CTW', recog='SAR', kie='SDMGR')

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

3.5 API Arguments

The API has an extensive list of arguments that you can use. The following tables are for the python interface.

MMOCR():

[1]: `kie` is only effective when both text detection and recognition models are specified.

注解： User can use default pretrained models by specifying `det` and/or `recog`, which is equivalent to specifying their corresponding `*_config` and `*_ckpt`. However, manually specifying `*_config` and `*_ckpt` will always override values set by `det` and/or `recog`. Similar rules also apply to `kie`, `kie_config` and `kie_ckpt`.

3.5.1 readtext():

[1]: Make sure that the model is compatible with batch mode.

[2]: Only effective when the script is running in det + recog mode.

All arguments are the same for the cli, all you need to do is add 2 hyphens at the beginning of the argument and replace underscores by hyphens. (*Example:* `det_batch_size` becomes `--det-batch-size`)

For bool type arguments, putting the argument in the command stores it as true. (*Example:* `python mmocr/utils/ocr.py demo/demo_text_det.jpg --batch_mode --print_result` means that `batch_mode` and `print_result` are set to `True`)

3.6 Models

Text detection:

Text recognition:

警告： SAR_CN is the only model that supports Chinese character recognition and it requires a Chinese dictionary. Please download the dictionary from here for a successful run.

Key information extraction:

3.7 Additional info

- To perform det + recog inference (end2end ocr), both the `det` and `recog` arguments must be defined.
- To perform only detection set the `recog` argument to `None`.
- To perform only recognition set the `det` argument to `None`.
- `details` argument only works with end2end ocr.
- `det_batch_size` and `recog_batch_size` arguments define the number of images you want to forward to the model at the same time. For maximum speed, set this to the highest number you can. The max batch size is limited by the model complexity and the GPU VRAM size.

If you have any suggestions for new features, feel free to open a thread or even PR :)

CHAPTER 4

Training

4.1 Training on a Single Machine

You can use `tools/train.py` to train a model in a single machine with one or more GPUs.

Here is the full usage of the script:

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

4.2 Training on Multiple Machines

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 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
```

4.3.1 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}
```

4.4 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.
```

(下页继续)

(续上页)

```
    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
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

Testing

We introduce the way to test pretrained models on datasets here.

5.1 Testing with Single GPU

You can use `tools/test.py` to perform single GPU inference. For example, to evaluate DBNet on IC15: (You can download pretrained models from *Model Zoo*):

```
./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
```

And here is the full usage of the script:

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

5.2 Testing with Multiple GPUs

MMOCR implements **distributed** testing with `MMDistributedDataParallel`.

You can use the following command to test a dataset with multiple GPUs.

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

For example,

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

5.3 Testing with Slurm

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

Here is an example of using 8 GPUs to test an example model on the ‘dev’ partition with job name ‘test_job’.

```
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 Batch Testing

By default, MMOCR tests the model image by image. For faster inference, you may change `data.val_dataloader.samples_per_gpu` and `data.test_dataloader.samples_per_gpu` in the config. For example,

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

will test the model with 16 images in a batch.

警告： Batch testing may incur performance decrease of the model due to the different behavior of the data preprocessing pipeline.

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

Model Serving

MMOCR provides some utilities that facilitate the model serving process. Here is a quick walkthrough of necessary steps that let the models to serve through an API.

7.1 Install TorchServe

You can follow the steps on the official website to install TorchServe and torch-model-archiver.

7.2 Convert model from MMOCR to TorchServe

We provide a handy tool to convert any .pth model into .mar model for TorchServe.

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

注解： ${MODEL_STORE} needs to be an absolute path to a folder.

For example:

```
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 Start Serving

7.3.1 From your Local Machine

Getting your models prepared, the next step is to start the service with a one-line command:

```
# To load all the models in ./checkpoints
torchserve --start --model-store ./checkpoints --models all
# Or, if you only want one model to serve, say dbnet
torchserve --start --model-store ./checkpoints --models dbnet=dbnet.mar
```

Then you can access inference, management and metrics services through TorchServe's REST API. You can find their usages in TorchServe REST API.

注解: By default, TorchServe binds port number `8080`, `8081` and `8082` to its services. You can change such behavior by modifying and saving the contents below to `config.properties`, and running TorchServe with option `--ts-config config.preperties`.

```
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 From Docker

A better alternative to serve your models is through Docker. We provide a Dockerfile that frees you from those tedious and error-prone environmental setup steps.

Build `mmocr-serve` Docker image

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

Run `mmocr-serve` with Docker

In order to run Docker in GPU, you need to install nvidia-docker; or you can omit the `--gpus` argument for a CPU-only session.

The command below will run `mmocr-serve` with a gpu, bind the ports of `8080` (inference), `8081` (management) and `8082` (metrics) from container to `127.0.0.1`, and mount the checkpoint folder `./checkpoints` from the host machine to `/home/model-server/model-store` of the container. For more information, please check the official docs for running TorchServe with docker.

```
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` points to the absolute path of "./checkpoints", and you can replace it with the absolute path where you store torchserve models.

Upon running the docker, you can access inference, management and metrics services through TorchServe's REST API. You can find their usages in TorchServe REST API.

7.4 4. Test deployment

Inference API allows user to post an image to a model and returns the prediction result.

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

For example,

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

For detection models, you should obtain a json with an object named `boundary_result`. Each array inside has float numbers representing x, y coordinates of boundary vertices in clockwise order, and the last float number as the confidence score.

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


For recognition models, the response should look like:


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


And you can use `test_torchserve.py` to compare result of TorchServe and PyTorch by visualizing them.

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

Example:

```
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

Dataset Types

8.1 General Introduction

To support the tasks of text detection, text recognition and key information extraction, we have designed some new types of dataset which consist of **loader** and **parser** to load and parse different types of annotation files.

- **loader**: Load the annotation file. There are two types of loader, `HardDiskLoader` and `LmdbLoader`
  - `HardDiskLoader`: Load `txt` format annotation file from hard disk to memory.
  - `LmdbLoader`: Load `lmdb` format annotation file with lmdb backend, which is very useful for **extremely large** annotation files to avoid out-of-memory problem when ten or more GPUs are used, since each GPU will start multiple processes to load annotation file to memory.
- **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.).

Here we show some examples of using different combination of `loader` and `parser`.

8.2 General Task

8.2.1 UniformConcatDataset

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

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

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

8.3 Text Detection Task

8.3.1 TextDetDataset

Dataset with annotation file in line-json txt format

```
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='HardDiskLoader',
        repeat=4,
        parser=dict(
            type='LineJsonParser',
            keys=['file_name', 'height', 'width', 'annotations'])),
    pipeline=test_pipeline,
    test_mode=True)
```

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]]}]}
```


8.3.2 IcdarDataset

Dataset with annotation file in coco-like json format

For text detection, you can also use an annotation file in a COCO format that is defined in MMDetection:

```
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)
```

You can check the content of the annotation file in `tests/data/toy_dataset/instances_test.json`.

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

8.4 Text Recognition Task

8.4.1 OCRDataset

Dataset for encoder-decoder based recognizer

```
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='HardDiskLoader',
        repeat=10,
        parser=dict(
            type='LineStrParser',
            keys=['filename', 'text'],
            keys_idx=[0, 1],
            separator=' ')),
    pipeline=train_pipeline,
    test_mode=False)
```

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'}`.

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`.

If the annotation file is extremely large, you can convert it from txt format to lmdb format with the following command:

```
python tools/data_converter/txt2lmdb.py -i ann_file.txt -o ann_file.lmdb
```

After that, you can use `LmdbLoader` in dataset like below.

```
img_prefix = 'tests/data/ocr_toy_dataset/imgs'
train_anno_file = 'tests/data/ocr_toy_dataset/label.lmdb'
train = dict(
    type=dataset_type,
    img_prefix=img_prefix,
    ann_file=train_anno_file,
```

(下页继续)

(续上页)

```
    loader=dict(
        type='LmdbLoader',
        repeat=10,
        parser=dict(
            type='LineStrParser',
            keys=['filename', 'text'],
            keys_idx=[0, 1],
            separator=' ')),
    pipeline=train_pipeline,
    test_mode=False)
```

8.4.2 OCRSegDataset

Dataset for segmentation-based recognizer

```
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='HardDiskLoader',
        repeat=10,
        parser=dict(
            type='LineJsonParser',
            keys=['file_name', 'annotations', 'text'])),
    pipeline=train_pipeline,
    test_mode=True)
```

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 9

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.

9.1 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.

9.2 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 .

9.2.1 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 10

概览

- 模型权重文件数量：29
- 配置文件数量：20
- 论文数量：16
  - ALGORITHM: 15
  - PREPROCESSOR: 1

10.1 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

10.2 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

10.3 文本检测模型

- 模型权重文件数量：14
- 配置文件数量：8
- 论文数量：7
  - [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
  - [ALGORITHM] Shape Robust Text Detection With Progressive Scale Expansion Network
  - [ALGORITHM] Textsnake: A Flexible Representation for Detecting Text of Arbitrary Shapes

10.4 文本识别模型

- 模型权重文件数量：11
- 配置文件数量：8
- 论文数量：7
  - [ALGORITHM] An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition
  - [ALGORITHM] Nrtr: A No-Recurrence Sequence-to-Sequence Model for Scene Text Recognition
  - [ALGORITHM] On Recognizing Texts of Arbitrary Shapes With 2d Self-Attention
  - [ALGORITHM] 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

- [PREPROCESSOR] Robust Scene Text Recognition With Automatic Rectification

CHAPTER 11

文本检测模型

11.1 Real-time Scene Text Detection with Differentiable Binarization

11.1.1 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.

11.1.2 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}}
```

11.1.3 Results and models

ICDAR2015

11.1.4 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.

11.1.5 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}
}
```

11.1.6 Results and models

CTW1500

Note: 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.

11.2 Fourier Contour Embedding for Arbitrary-Shaped Text Detection

11.2.1 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.

11.2.2 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}
      }
```

11.2.3 Results and models

CTW1500

ICDAR2015

11.2.4 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.

11.2.5 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}}
```

In tuning parameters, we refer to the baseline method in the following article:

```
@article{pmtd,
  author={Jingchao Liu and Xuebo Liu and Jie Sheng and Ding Liang and Xin Li and
↪Qingjie Liu},
  title={Pyramid Mask Text Detector},
  journal={CoRR},
  volume={abs/1903.11800},
  year={2019}
}
```

11.2.6 Results and models

CTW1500

ICDAR2015

ICDAR2017

11.2.7 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.

11.2.8 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}
  }
```

11.2.9 Results and models

CTW1500

ICDAR2015

Note: 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.

11.3 PSENet

11.3.1 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.

11.3.2 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}
}
```

11.3.3 Results and models

CTW1500

ICDAR2015

Note: 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.

11.4 Textsnake

11.4.1 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.

11.4.2 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}
}
```

11.4.3 Results and models

CTW1500

CHAPTER 12

文本识别模型

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

12.1.1 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.

12.1.2 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}
}
```

12.1.3 Dataset

Train Dataset

Test Dataset

12.1.4 Results and models

12.1.5 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).

12.1.6 Citation

Main paper

```
@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)},
```

(下页继续)

(续上页)

```
  pages={781--786},
  year={2019},
  organization={IEEE}
}
```

Backbone

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

12.1.7 Dataset

Train Dataset

Test Dataset

12.1.8 Results and Models

Notes:

- 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.

12.2 RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

12.2.1 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.

12.2.2 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}
}
```

12.2.3 Dataset

Train Dataset

Test Dataset

12.2.4 Results and Models

12.2.5 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.

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

12.3.1 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.

12.3.2 Citation

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

12.3.3 Dataset

Train Dataset

Test Dataset

12.3.4 Results and Models

12.3.5 Chinese Dataset

12.3.6 Results and Models

Notes:

- `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.

12.3.7 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.

12.4 SATRN

12.4.1 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.

12.4.2 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}
}
```

12.4.3 Dataset

Train Dataset

Test Dataset

12.4.4 Results and Models

12.4.5 Abstract

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

12.4.6 Citation

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

12.4.7 Dataset

Train Dataset

Test Dataset

12.4.8 Results and Models

Notes:

- `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.

12.5 CRNN with TPS based STN

12.5.1 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.

12.5.2 Citation

Main paper

```
@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}
}
```

Preprocessor

```
@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}
}
```

12.5.3 Dataset

Train Dataset

Test Dataset

12.5.4 Results and models

CHAPTER 13

关键信息提取模型

13.1 Spatial Dual-Modality Graph Reasoning for Key Information Extraction

13.1.1 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.

13.1.2 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}
}
```

13.1.3 Results and models

WildReceipt

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*.

CHAPTER 14

命名实体识别模型

14.1 Chinese Named Entity Recognition using BERT + Softmax

14.1.1 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).

14.1.2 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},
```

(下页继续)

(续上页)

```
  year={2018}
}
```

14.1.3 Dataset

Train Dataset

Test Dataset

14.1.4 Results and models

CHAPTER 15

文字检测

15.1 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
```

15.2 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）

15.3 准备步骤

15.3.1 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
```

15.3.2 ICDAR 2017

- 与上述步骤类似。

15.3.3 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
```

15.3.4 SynthText

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

15.3.5 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
```

15.3.6 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 16

文字识别

16.1 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
|   |   ├── 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
|   |   ├── train_1_label.txt
```

(下页继续)

(续上页)

```
|   |   ├── 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。

16.2 准备步骤

16.2.1 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

16.2.2 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

16.2.3 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

16.2.4 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>
```

16.2.5 ct80

- 第一步：下载 test_label.txt

16.2.6 svtp

- 第一步：下载 test_label.txt

16.2.7 coco_text

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

16.2.8 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
```

16.2.9 SynthText (Synth800k)

- 第一步：从 下载地址 下载 `SynthText.zip`
- 第二步：

```
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 .

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

- 第三步：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
```

16.2.10 SynthAdd

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

```
mkdir SynthAdd && cd SynthAdd

mv /path/to/SynthText_Add.zip .

unzip SynthText_Add.zip

mv /path/to/label.txt .

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

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
```

16.2.11 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
```

16.2.12 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
```

(下页继续)

(续上页)

```
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
```

16.2.13 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
  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
```

(下页继续)

(续上页)

```
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 17

关键信息提取

17.1 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
```

17.2 准备步骤

17.2.1 WildReceipt

- 下载并解压 wildreceipt.tar

17.2.2 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 18

命名实体识别（专名识别）

18.1 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
```

18.2 准备步骤

18.2.1 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 19

Useful Tools

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

19.1 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`.

19.2 Convert txt annotation to lmdb format

Sometimes, loading a large txt annotation file with multiple workers can cause OOM (out of memory) error. You can convert the file into lmdb format using `tools/data/utils/txt2lmdb.py` and use LmdbLoader in your config to avoid this issue.

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

For example,

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

CHAPTER 20

Changelog

20.1 v0.3.0 (25/8/2021)

20.1.1 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

20.1.2 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.

20.1.3 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

20.1.4 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

20.1.5 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

20.1.6 Contributors

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

20.2 v0.2.1 (20/7/2021)

20.2.1 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

20.2.2 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

20.2.3 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

20.2.4 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

20.3 v0.2.0 (18/5/2021)

20.3.1 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.

20.3.2 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

20.3.3 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

20.3.4 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

20.4 v0.1.0 (7/4/2021)

20.4.1 Highlights

- MMOCR is released.

20.4.2 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 21

mmocr.apis

mmocr.apis.**init_detector**(*config*, *checkpoint=None*, *device='cuda:0'*, *cfg_options=None*)

Initialize a detector from config file.

参数

- **config** (str or mmcv.Config) –Config file path or the config object.
- **checkpoint** (*str, optional*) –Checkpoint path. If left as None, the model will not load any weights.
- **cfg_options** (*dict*) –Options to override some settings in the used config.

返回 The constructed detector.

返回类型 nn.Module

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

Initialize random seed. If the seed is None, it will be replaced by a random number, and then broadcasted to all processes.

参数

- **seed** (*int, Optional*) –The seed.
- **device** (*str*) –The device where the seed will be put on.

返回 Seed to be used.

返回类型 int

mmocr.apis.**model_inference**(*model*, *imgs*, *ann=None*, *batch_mode=False*, *return_data=False*)

Inference image(s) with the detector.

参数

- **model** (*nn.Module*) –The loaded detector.
- **imgs** (*str/ndarray or list[str/ndarray] or tuple[str/ndarray]*) – Either image files or loaded images.
- **batch_mode** (*bool*) –If True, use batch mode for inference.
- **ann** (*dict*) –Annotation info for key information extraction.
- **return_data** –Return postprocessed data.

返回 Predicted results.

返回类型 result (dict)

CHAPTER 22

mmocr.core

22.1 evaluation

mmocr.core.evaluation.**compute_f1_score**(*preds*, *gts*, *ignores=[]*)

Compute the F1-score of prediction.

参数

- **preds** (*Tensor*) –The predicted probability NxC map with N and C being the sample number and class number respectively.
- **gts** (*Tensor*) –The ground truth vector of size N.
- **ignores** –The index set of classes that are ignored when reporting results. Note: all samples are participated in computing.

mmocr.core.evaluation.**eval_hmean**(*results*, *img_infos*, *ann_infos*, *metrics={'hmean-iou'}*, *score_thr=0.3*, *rank_list=None*, *logger=None*, ***kwargs*)

Evaluation in hmean metric.

参数

- **results** (*list[dict]*) –Each dict corresponds to one image, containing the following keys: boundary_result
- **img_infos** (*list[dict]*) –Each dict corresponds to one image, containing the following keys: filename, height, width

- **ann_infos** (*list[dict]*) –Each dict corresponds to one image, containing the following keys: masks, masks_ignore
- **score_thr** (*float*) –Score threshold of prediction map.
- **metrics** (*set{str}*) –Hmean metric set, should be one or all of { 'hmean-iou', 'hmean-ic13' }

返回 float]

返回类型 dict[str

mmocr.core.evaluation.**eval_hmean_ic13**(*det_boxes*, *gt_boxes*, *gt_ignored_boxes*, *precision_thr=0.4*, *recall_thr=0.8*, *center_dist_thr=1.0*, *one2one_score=1.0*, *one2many_score=0.8*, *many2one_score=1.0*)

Evaluate hmean of text detection using the icdar2013 standard.

参数

- **det_boxes** (*list[list[list[float]]]*) –List of arrays of shape (n, 2k). Each element is the det_boxes for one img. k>=4.
- **gt_boxes** (*list[list[list[float]]]*) –List of arrays of shape (m, 2k). Each element is the gt_boxes for one img. k>=4.
- **gt_ignored_boxes** (*list[list[list[float]]]*) –List of arrays of (l, 2k). Each element is the ignored gt_boxes for one img. k>=4.
- **precision_thr** (*float*) –Precision threshold of the iou of one (gt_box, det_box) pair.
- **recall_thr** (*float*) –Recall threshold of the iou of one (gt_box, det_box) pair.
- **center_dist_thr** (*float*) –Distance threshold of one (gt_box, det_box) center point pair.
- **one2one_score** (*float*) –Reward when one gt matches one det_box.
- **one2many_score** (*float*) –Reward when one gt matches many det_boxes.
- **many2one_score** (*float*) –Reward when many gts match one det_box.

返回 Tuple of dicts which encodes the hmean for the dataset and all images.

返回类型 hmean (tuple[dict])

mmocr.core.evaluation.**eval_hmean_iou**(*pred_boxes*, *gt_boxes*, *gt_ignored_boxes*, *iou_thr=0.5*, *precision_thr=0.5*)

Evaluate hmean of text detection using IOU standard.

参数

- **pred_boxes** (*list[list[list[float]]]*) –Text boxes for an img list. Each box has 2k (>=8) values.

- **gt_boxes** (*list[list[list[float]]]*) –Ground truth text boxes for an img list. Each box has 2k (>=8) values.
- **gt_ignored_boxes** (*list[list[list[float]]]*) –Ignored ground truth text boxes for an img list. Each box has 2k (>=8) values.
- **iou_thr** (*float*) –Iou threshold when one (gt_box, det_box) pair is matched.
- **precision_thr** (*float*) –Precision threshold when one (gt_box, det_box) pair is matched.

返回

Tuple of dicts indicates the hmean for the dataset and all images.

返回类型 hmean (tuple[dict])

mmocr.core.evaluation.**eval_ner_f1**(*results*, *gt_infos*)

Evaluate for ner task.

参数

- **results** (*list*) –Predict results of entities.
- **gt_infos** (*list[dict]*) –Ground-truth information which contains text and label.

返回

precision,recall, f1-score of total and each catogory.

返回类型 class_info (dict)

mmocr.core.evaluation.**eval_ocr_metric**(*pred_texts*, *gt_texts*)

Evaluate the text recognition performance with metric: word accuracy and 1-N.E.D. See https://rrc.cvc.uab.es/?ch=14&com=tasks for details.

参数

- **pred_texts** (*list[str]*) –Text strings of prediction.
- **gt_texts** (*list[str]*) –Text strings of ground truth.

返回

float]): Metric dict for text recognition, include:

- word_acc: Accuracy in word level.
- word_acc_ignore_case: Accuracy in word level, ignore letter case.
- **word_acc_ignore_case_symbol: Accuracy in word level, ignore** letter case and symbol. (default metric for academic evaluation)
- **char_recall: Recall in character level, ignore** letter case and symbol.
- **char_precision: Precision in character level, ignore** letter case and symbol.

- 1-N.E.D: 1 - normalized_edit_distance.

返回类型 eval_res (dict[str

CHAPTER 23

mmocr.utils

class mmocr.utils.**Registry**(*name*, *build_func=None*, *parent=None*, *scope=None*)

A registry to map strings to classes.

Registered object could be built from registry. .. rubric:: 示例

```
>>> MODELS = Registry('models')
>>> @MODELS.register_module()
>>> class ResNet:
>>>     pass
>>> resnet = MODELS.build(dict(type='ResNet'))
```

Please refer to https://mmcv.readthedocs.io/en/latest/understand_mmcv/registry.html for advanced usage.

参数

- **name** (*str*) –Registry name.
- **build_func** (*func, optional*) –Build function to construct instance from Registry, func:*build_from_cfg* is used if neither `parent` or `build_func` is specified. If `parent` is specified and `build_func` is not given, `build_func` will be inherited from `parent`. Default: None.
- **parent** (Registry, *optional*) –Parent registry. The class registered in children registry could be built from parent. Default: None.
- **scope** (*str, optional*) –The scope of registry. It is the key to search for children registry. If not specified, scope will be the name of the package where class is defined, e.g. mmdet, mmcls, mmseg. Default: None.

get(*key*)

Get the registry record.

参数 **key** (*str*) –The class name in string format.

返回 The corresponding class.

返回类型 class

static infer_scope()

Infer the scope of registry.

The name of the package where registry is defined will be returned.

示例

in mmdet/models/backbone/resnet.py »> MODELS = Registry(‘models’) »> @MODELS.register_module() »> class ResNet: »> pass The scope of `ResNet` will be `mmdet`.

返回 The inferred scope name.

返回类型 scope (str)

register_module(*name=None*, *force=False*, *module=None*)

Register a module.

A record will be added to *self._module_dict*, whose key is the class name or the specified name, and value is the class itself. It can be used as a decorator or a normal function.

示例

```
>>> backbones = Registry('backbone')
>>> @backbones.register_module()
>>> class ResNet:
>>>     pass
```

```
>>> backbones = Registry('backbone')
>>> @backbones.register_module(name='mnet')
>>> class MobileNet:
>>>     pass
```

```
>>> backbones = Registry('backbone')
>>> class ResNet:
>>>     pass
>>> backbones.register_module(ResNet)
```

参数

- **name** (*str | None*) –The module name to be registered. If not specified, the class name will be used.
- **force** (*bool, optional*) –Whether to override an existing class with the same name. Default: False.
- **module** (*type*) –Module class to be registered.

static split_scope_key(*key*)

Split scope and key.

The first scope will be split from key.

实际案例

```
>>> Registry.split_scope_key('mmdet.ResNet')
'mmdet', 'ResNet'
>>> Registry.split_scope_key('ResNet')
None, 'ResNet'
```

返回 The first scope. key (str): The remaining key.

返回类型 scope (str, None)

class mmocr.utils.**StringStrip**(*strip=True*, *strip_pos='both'*, *strip_str=None*)

Removing the leading and/or the trailing characters based on the string argument passed.

参数

- **strip** (*bool*) –Whether remove characters from both left and right of the string. Default: True.
- **strip_pos** (*str*) –Which position for removing, can be one of ('both', 'left', 'right'), Default: 'both'.
- **strip_str** (*str|None*) –A string specifying the set of characters to be removed from the left and right part of the string. If None, all leading and trailing whitespaces are removed from the string. Default: None.

mmocr.utils.**build_from_cfg**(*cfg*, *registry*, *default_args=None*)

Build a module from config dict.

参数

- **cfg** (*dict*) –Config dict. It should at least contain the key "type".
- **registry** (*Registry*) –The registry to search the type from.
- **default_args** (*dict, optional*) –Default initialization arguments.

返回 The constructed object.

返回类型 object

mmocr.utils.**collect_env**()

Collect the information of the running environments.

mmocr.utils.**convert_annotations**(*image_infos*, *out_json_name*)

Convert the annotation into coco style.

参数

- **image_infos** (*list*) –The list of image information dicts
- **out_json_name** (*str*) –The output json filename

返回 The coco style dict

返回类型 out_json(dict)

mmocr.utils.**drop_orientation**(*img_file*)

Check if the image has orientation information. If yes, ignore it by converting the image format to png, and return new filename, otherwise return the original filename.

参数 **img_file** (*str*) –The image path

返回 The converted image filename with proper postfix

mmocr.utils.**get_root_logger**(*log_file=None*, *log_level=20*)

Use *get_logger* method in mmcv to get the root logger.

The logger will be initialized if it has not been initialized. By default a StreamHandler will be added. If *log_file* is specified, a FileHandler will also be added. The name of the root logger is the top-level package name, e.g., "mmpose".

参数

- **log_file** (*str | None*) –The log filename. If specified, a FileHandler will be added to the root logger.
- **log_level** (*int*) –The root logger level. Note that only the process of rank 0 is affected, while other processes will set the level to "Error" and be silent most of the time.

返回 The root logger.

返回类型 logging.Logger

mmocr.utils.**is_2dlist**(*x*)

check x is 2d-list([[1], []]) or 1d empty list([]).

Notice: The reason that it contains 1d empty list is because some arguments from gt annotation file or model prediction may be empty, but usually, it should be 2d-list.

mmocr.utils.**is_3dlist**(*x*)

check x is 3d-list([[[1], []]]) or 2d empty list([[], []]) or 1d empty list([]).

Notice: The reason that it contains 1d or 2d empty list is because some arguments from gt annotation file or model prediction may be empty, but usually, it should be 3d-list.

mmocr.utils.**is_not_png**(*img_file*)

Check img_file is not png image.

参数 **img_file** (*str*) –The input image file name

返回 The bool flag indicating whether it is not png

mmocr.utils.**is_on_same_line**(*box_a*, *box_b*, *min_y_overlap_ratio=0.8*)

Check if two boxes are on the same line by their y-axis coordinates.

Two boxes are on the same line if they overlap vertically, and the length of the overlapping line segment is greater than min_y_overlap_ratio * the height of either of the boxes.

参数

- **box_a** (*list), box_b (list*) –Two bounding boxes to be checked
- **min_y_overlap_ratio** (*float*) –The minimum vertical overlapping ratio allowed for boxes in the same line

返回 The bool flag indicating if they are on the same line

mmocr.utils.**list_from_file**(*filename*, *encoding='utf-8'*)

Load a text file and parse the content as a list of strings. The trailing "r" and "n" of each line will be removed.

注解: This will be replaced by mmcv's version after it supports encoding.

参数

- **filename** (*str*) –Filename.
- **encoding** (*str*) –Encoding used to open the file. Default utf-8.

返回 A list of strings.

返回类型 list[str]

mmocr.utils.**list_to_file**(*filename*, *lines*)

Write a list of strings to a text file.

参数

- **filename** (*str*) –The output filename. It will be created/overwritten.
- **lines** (*list(str)*) –Data to be written.

mmocr.utils.**revert_sync_batchnorm**(*module*)

Helper function to convert all *SyncBatchNorm* layers in the model to *BatchNormXd* layers.

Adapted from @kapily’s work: (https://github.com/pytorch/pytorch/issues/41081#issuecomment-783961547)

参数 **module** (*nn.Module*) –The module containing *SyncBatchNorm* layers.

返回 The converted module with *BatchNormXd* layers.

返回类型 module_output

mmocr.utils.**stitch_boxes_into_lines**(*boxes*, *max_x_dist=10*, *min_y_overlap_ratio=0.8*)

Stitch fragmented boxes of words into lines.

Note: part of its logic is inspired by @Johndirr (https://github.com/faustomorales/keras-ocr/issues/22)

参数

- **boxes** (*list*) –List of ocr results to be stitched
- **max_x_dist** (*int*) –The maximum horizontal distance between the closest edges of neighboring boxes in the same line
- **min_y_overlap_ratio** (*float*) –The minimum vertical overlapping ratio allowed for any pairs of neighboring boxes in the same line

返回 List of merged boxes and texts

返回类型 merged_boxes(list[dict])

CHAPTER 24

mmocr.models

24.1 common_backbones

class mmocr.models.common.backbones.**UNet**(*in_channels=3*, *base_channels=64*, *num_stages=5*, *strides=(1, 1, 1, 1, 1)*, *enc_num_convs=(2, 2, 2, 2, 2)*, *dec_num_convs=(2, 2, 2, 2)*, *downsamples=(True, True, True, True)*, *enc_dilations=(1, 1, 1, 1, 1)*, *dec_dilations=(1, 1, 1, 1)*, *with_cp=False*, *conv_cfg=None*, *norm_cfg={'type': 'BN'}*, *act_cfg={'type': 'ReLU'}*, *upsample_cfg={'type': 'InterpConv'}*, *norm_eval=False*, *dcn=None*, *plugins=None*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Constant', 'layer': ['_BatchNorm', 'GroupNorm'], 'val': 1}]*)

UNet backbone. U-Net: Convolutional Networks for Biomedical Image Segmentation. https://arxiv.org/pdf/1505.04597.pdf

参数

- **in_channels** (*int*) –Number of input image channels. Default" 3.
- **base_channels** (*int*) –Number of base channels of each stage. The output channels of the first stage. Default: 64.
- **num_stages** (*int*) –Number of stages in encoder, normally 5. Default: 5.

- **strides** (*Sequence[int 1 | 2]*) –Strides of each stage in encoder. len(strides) is equal to num_stages. Normally the stride of the first stage in encoder is 1. If strides[i]=2, it uses stride convolution to downsample in the correspondence encoder stage. Default: (1, 1, 1, 1, 1).
- **enc_num_convs** (*Sequence[int]*) –Number of convolutional layers in the convolution block of the correspondence encoder stage. Default: (2, 2, 2, 2, 2).
- **dec_num_convs** (*Sequence[int]*) –Number of convolutional layers in the convolution block of the correspondence decoder stage. Default: (2, 2, 2, 2).
- **downsamples** (*Sequence[int]*) –Whether use MaxPool to downsample the feature map after the first stage of encoder (stages: [1, num_stages)). If the correspondence encoder stage use stride convolution (strides[i]=2), it will never use MaxPool to downsample, even downsamples[i-1]=True. Default: (True, True, True, True).
- **enc_dilations** (*Sequence[int]*) –Dilation rate of each stage in encoder. Default: (1, 1, 1, 1, 1).
- **dec_dilations** (*Sequence[int]*) –Dilation rate of each stage in decoder. Default: (1, 1, 1, 1).
- **with_cp** (*bool*) –Use checkpoint or not. Using checkpoint will save some memory while slowing down the training speed. Default: False.
- **conv_cfg** (*dict | None*) –Config dict for convolution layer. Default: None.
- **norm_cfg** (*dict | None*) –Config dict for normalization layer. Default: dict(type='BN').
- **act_cfg** (*dict | None*) –Config dict for activation layer in ConvModule. Default: dict(type='ReLU').
- **upsample_cfg** (*dict*) –The upsample config of the upsample module in decoder. Default: dict(type='InterpConv').
- **norm_eval** (*bool*) –Whether to set norm layers to eval mode, namely, freeze running stats (mean and var). Note: Effect on Batch Norm and its variants only. Default: False.
- **dcn** (*bool*) –Use deformable convolution in convolutional layer or not. Default: None.
- **plugins** (*dict*) –plugins for convolutional layers. Default: None.

Notice: The input image size should be divisible by the whole downsample rate of the encoder. More detail of the whole downsample rate can be found in UNet._check_input_divisible.

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

train(*mode=True*)

Convert the model into training mode while keep normalization layer freezed.

class mmocr.models.common.losses.**DiceLoss**(*eps=1e-06*)

forward(*pred*, *target*, *mask=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

class mmocr.models.common.losses.**FocalLoss**(*gamma=2*, *weight=None*, *ignore_index=- 100*)

Multi-class Focal loss implementation.

参数

- **gamma** (*float*) –The larger the gamma, the smaller the loss weight of easier samples.
- **weight** (*float*) –A manual rescaling weight given to each class.
- **ignore_index** (*int*) –Specifies a target value that is ignored and does not contribute to the input gradient.

forward(*input*, *target*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

24.2 textdet_dense_heads

class mmocr.models.textdet.dense_heads.**DBHead**(*in_channels*, *with_bias=False*, *decoding_type='db'*, *text_repr_type='poly'*, *downsample_ratio=1.0*, *loss={'type': 'DBLoss'}*, *train_cfg=None*, *test_cfg=None*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv'}, {'type': 'Constant', 'layer': 'BatchNorm', 'val': 1.0, 'bias': 0.0001}]*)

The class for DBNet head.

This was partially adapted from https://github.com/MhLiao/DB

参数

- **in_channels** (*int*) –The number of input channels of the db head.
- **decoding_type** (*str*) –The type of decoder for dbnet.
- **text_repr_type** (*str*) –Boundary encoding type 'poly' or 'quad'.
- **downsample_ratio** (*float*) –The downsample ratio of ground truths.
- **loss** (*dict*) –The type of loss for dbnet.

forward(*inputs*)

参数 **inputs** (*Tensor*) –Shape (batch_size, hidden_size, h, w).

返回 A tensor of the same shape as input.

返回类型 Tensor

class mmocr.models.textdet.dense_heads.**DRRGHead**(*in_channels*, *k_at_hops=(8, 4)*, *num_adjacent_linkages=3*, *node_geo_feat_len=120*, *pooling_scale=1.0*, *pooling_output_size=(4, 3)*, *nms_thr=0.3*, *min_width=8.0*, *max_width=24.0*, *comp_shrink_ratio=1.03*, *comp_ratio=0.4*, *comp_score_thr=0.3*, *text_region_thr=0.2*, *center_region_thr=0.2*, *center_region_area_thr=50*, *local_graph_thr=0.7*, *link_thr=0.85*, *loss={'type': 'DRRGLoss'}*, *train_cfg=None*, *test_cfg=None*, *init_cfg={'mean': 0, 'override': {'name': 'out_conv'}, 'std': 0.01, 'type': 'Normal'}*)

The class for DRRG head: Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection.

参数

- **k_at_hops** (*tuple(int)*) –The number of i-hop neighbors, i = 1, 2.
- **num_adjacent_linkages** (*int*) –The number of linkages when constructing adjacent matrix.
- **node_geo_feat_len** (*int*) –The length of embedded geometric feature vector of a component.
- **pooling_scale** (*float*) –The spatial scale of rotated RoI-Align.
- **pooling_output_size** (*tuple(int)*) –The output size of RRoI-Aligning.
- **nms_thr** (*float*) –The locality-aware NMS threshold of text components.
- **min_width** (*float*) –The minimum width of text components.
- **max_width** (*float*) –The maximum width of text components.
- **comp_shrink_ratio** (*float*) –The shrink ratio of text components.
- **comp_ratio** (*float*) –The reciprocal of aspect ratio of text components.
- **comp_score_thr** (*float*) –The score threshold of text components.
- **text_region_thr** (*float*) –The threshold for text region probability map.
- **center_region_thr** (*float*) –The threshold for text center region probability map.
- **center_region_area_thr** (*int*) –The threshold for filtering small-sized text center region.
- **local_graph_thr** (*float*) –The threshold to filter identical local graphs.
- **link_thr** (*float*) –The threshold for connected components search.
- **loss** (*dict*) –The config of loss that DRRGHead uses.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*inputs*, *gt_comp_attribs*)

参数

- **inputs** (*Tensor*) –Shape of (N, C, H, W).
- **gt_comp_attribs** (*list[ndarray]*) –The padded text component attributes. Shape: (num_component, 8).

返回

Returns (pred_maps, (gcn_pred, gt_labels)).

- pred_maps (Tensor): Prediction map with shape (N, C_{out}, H, W).
- gcn_pred (Tensor): Prediction from GCN module, with shape $(N, 2)$.

- gt_labels (Tensor): Ground-truth label with shape $(N, 8)$.

返回类型 tuple

get_boundary(*edges*, *scores*, *text_comps*, *img_metas*, *rescale*)

Compute text boundaries via post processing.

参数

- **edges** (*ndarray*) –The edge array of shape N * 2, each row is a pair of text component indices that makes up an edge in graph.
- **scores** (*ndarray*) –The edge score array.
- **text_comps** (*ndarray*) –The text components.
- **img_metas** (*list[dict]*) –The image meta infos.
- **rescale** (*bool*) –Rescale boundaries to the original image resolution.

返回 The result dict containing key *boundary_result*.

返回类型 dict

single_test(*feat_maps*)

参数 **feat_maps** (*Tensor*) –Shape of (N, C, H, W).

返回

Returns (edge, score, text_comps).

- edge (ndarray): The edge array of shape $(N, 2)$ where each row is a pair of text component indices that makes up an edge in graph.
- score (ndarray): The score array of shape $(N,)$, corresponding to the edge above.
- text_comps (ndarray): The text components of shape $(N, 9)$ where each row corresponds to one box and its score: (x1, y1, x2, y2, x3, y3, x4, y4, score).

返回类型 tuple

class mmocr.models.textdet.dense_heads.**FCEHead**(*in_channels*, *scales*, *fourier_degree=5*, *num_sample=50*, *num_reconstr_points=50*, *decoding_type='fcenet'*, *loss={'type': 'FCELoss'}*, *score_thr=0.3*, *nms_thr=0.1*, *alpha=1.0*, *beta=1.0*, *text_repr_type='poly'*, *train_cfg=None*, *test_cfg=None*, *init_cfg={'mean': 0, 'override': [{'name': 'out_conv_cls'}, {'name': 'out_conv_reg'}], 'std': 0.01, 'type': 'Normal'}*)

The class for implementing FCENet head.

FCENet(CVPR2021): Fourier Contour Embedding for Arbitrary-shaped Text Detection

参数

- **in_channels** (*int*) –The number of input channels.
- **scales** (*list[int]*) –The scale of each layer.
- **fourier_degree** (*int*) –The maximum Fourier transform degree k.
- **num_sample** (*int*) –The sampling points number of regression loss. If it is too small, FCEnet tends to be overfitting.
- **score_thr** (*float*) –The threshold to filter out the final candidates.
- **nms_thr** (*float*) –The threshold of nms.
- **alpha** (*float*) –The parameter to calculate final scores. $Score_{final} = (Score^{\alpha}_{text\_region}) * (Score^{\beta}_{text\_center\_region})$
- **beta** (*float*) –The parameter to calculate final scores.

forward(*feats*)

参数 **feats** (*list[Tensor]*) –Each tensor has the shape of (N, C_i, H_i, W_i).

返回 Each pair of tensors corresponds to the classification result and regression result computed from the input tensor with the same index. They have the shapes of $(N, C_{cls,i}, H_i, W_i)$ and $(N, C_{out,i}, H_i, W_i)$.

返回类型 list[[Tensor, Tensor]]

get_boundary(*score_maps*, *img_metas*, *rescale*)

Compute text boundaries via post processing.

参数

- **score_maps** (*Tensor*) –The text score map.
- **img_metas** (*dict*) –The image meta info.
- **rescale** (*bool*) –Rescale boundaries to the original image resolution if true, and keep the score_maps resolution if false.

返回 A dict where boundary results are stored in `boundary_result`.

返回类型 dict

class mmocr.models.textdet.dense_heads.**HeadMixin**

The head minxin for dbnet and pannet heads.

get_boundary(*score_maps*, *img_metas*, *rescale*)

Compute text boundaries via post processing.

参数

- **score_maps** (*Tensor*) –The text score map.
- **img_metas** (*dict*) –The image meta info.
- **rescale** (*bool*) –Rescale boundaries to the original image resolution if true, and keep the score_maps resolution if false.

返回 A dict where boundary results are stored in `boundary_result`.

返回类型 dict

loss(*pred_maps*, ***kwargs*)

Compute the loss for text detection.

参数 **pred_maps** (*Tensor*) –The input score maps of shape $(NxCxHxW)$.

返回 The dict for losses.

返回类型 dict

resize_boundary(*boundaries*, *scale_factor*)

Rescale boundaries via scale_factor.

参数

- **boundaries** (*list[list[float]]*) –The boundary list. Each boundary has $2k+1$ elements with $k >= 4$.
- **scale_factor** (*ndarray*) –The scale factor of size $(4,)$.

返回 The scaled boundaries.

返回类型 list[list[float]]

class mmocr.models.textdet.dense_heads.**PANHead**(*in_channels*, *out_channels*, *text_repr_type='poly'*, *downsample_ratio=0.25*, *loss={'type': 'PANLoss'}*, *train_cfg=None*, *test_cfg=None*, *init_cfg={'mean': 0, 'override': {'name': 'out_conv'}, 'std': 0.01, 'type': 'Normal'}*)

The class for PANet head.

参数

- **in_channels** (*list[int]*) –A list of 4 numbers of input channels.
- **out_channels** (*int*) –Number of output channels.
- **text_repr_type** (*str*) –Use polygon or quad to represent. Available options are "poly" or "quad".
- **downsample_ratio** (*float*) –Downsample ratio.

- **loss** (*dict*) –Configuration dictionary for loss type. Supported loss types are "PANLoss" and "PSELoss".
- **train_cfg** (*dict*) –Depreciated.
- **test_cfg** (*dict*) –Depreciated.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*inputs*)

参数 **inputs** (*list[Tensor] | Tensor*) –Each tensor has the shape of (N, C_i, W, H), where $\sum_i C_i = C_{in}$ and C_{in} is `input_channels`.

返回 A tensor of shape (N, C_{out}, W, H) where C_{out} is `output_channels`.

返回类型 Tensor

class mmocr.models.textdet.dense_heads.**PSEHead**(*in_channels*, *out_channels*, *text_repr_type='poly'*, *downsample_ratio=0.25*, *loss={'type': 'PSELoss'}*, *train_cfg=None*, *test_cfg=None*, *init_cfg=None*)

The class for PSENet head.

参数

- **in_channels** (*list[int]*) –A list of 4 numbers of input channels.
- **out_channels** (*int*) –Number of output channels.
- **text_repr_type** (*str*) –Use polygon or quad to represent. Available options are "poly" or "quad".
- **downsample_ratio** (*float*) –Downsample ratio.
- **loss** (*dict*) –Configuration dictionary for loss type. Supported loss types are "PANLoss" and "PSELoss".
- **train_cfg** (*dict*) –Depreciated.
- **test_cfg** (*dict*) –Depreciated.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

class mmocr.models.textdet.dense_heads.**TextSnakeHead**(*in_channels*, *decoding_type='textsnake'*, *text_repr_type='poly'*, *loss={'type': 'TextSnakeLoss'}*, *train_cfg=None*, *test_cfg=None*, *init_cfg={'mean': 0, 'override': {'name': 'out_conv'}, 'std': 0.01, 'type': 'Normal'}*)

The class for TextSnake head.

TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

参数

- **in_channels** (*int*) –Number of input channels.
- **decoding_type** (*str*) –Decoding type. It usually should not be changed.
- **text_repr_type** (*str*) –Use polygon or quad to represent. Available options are "poly" or "quad".
- **loss** (*dict*) –Configuration dictionary for loss type.
- **train_cfg** –Depreciated.
- **test_cfg** –Depreciated.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*inputs*)

参数 **inputs** (*Tensor*) –Shape (N, C_{in}, H, W), where C_{in} is `in_channels`. H and W should be the same as the input of backbone.

返回 A tensor of shape $(N, 5, H, W)$.

返回类型 Tensor

24.3 textdet_necks

class mmocr.models.textdet.necks.**FPEM_FFM**(*in_channels*, *conv_out=128*, *fpem_repeat=2*, *align_corners=False*, *init_cfg={'distribution': 'uniform', 'layer': 'Conv2d', 'type': 'Xavier'}*)

This code is from https://github.com/WenmuZhou/PAN.pytorch.

参数

- **in_channels** (*list[int]*) –A list of 4 numbers of input channels.
- **conv_out** (*int*) –Number of output channels.
- **fpem_repeat** (*int*) –Number of FPEM layers before FFM operations.
- **align_corners** (*bool*) –The interpolation behaviour in FFM operation, used in `torch.nn.functional.interpolate()`.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*x*)

参数 **x** (*list[Tensor]*) –A list of four tensors of shape (N, C_i, H_i, W_i), representing C2, C3, C4, C5 features respectively. C_i should matches the number in `in_channels`.

返回 Four tensors of shape (N, C_{out}, H_0, W_0) where C_{out} is `conv_out`.

返回类型 list[Tensor]

class mmocr.models.textdet.necks.**FPNC**(*in_channels*, *lateral_channels=256*, *out_channels=64*, *bias_on_lateral=False*, *bn_re_on_lateral=False*, *bias_on_smooth=False*, *bn_re_on_smooth=False*, *conv_after_concat=False*, *init_cfg=None*)

FPN-like fusion module in Real-time Scene Text Detection with Differentiable Binarization.

This was partially adapted from https://github.com/MhLiao/DB and https://github.com/WenmuZhou/DBNet.pytorch.

参数

- **in_channels** (*list[int]*) –A list of numbers of input channels.
- **lateral_channels** (*int*) –Number of channels for lateral layers.
- **out_channels** (*int*) –Number of output channels.
- **bias_on_lateral** (*bool*) –Whether to use bias on lateral convolutional layers.
- **bn_re_on_lateral** (*bool*) –Whether to use BatchNorm and ReLU on lateral convolutional layers.
- **bias_on_smooth** (*bool*) –Whether to use bias on smoothing layer.
- **bn_re_on_smooth** (*bool*) –Whether to use BatchNorm and ReLU on smoothing layer.
- **conv_after_concat** (*bool*) –Whether to add a convolution layer after the concatenation of predictions.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*inputs*)

参数 **inputs** (*list[Tensor]*) –Each tensor has the shape of (N, C_i, H_i, W_i). It usually expects 4 tensors (C2-C5 features) from ResNet.

返回 A tensor of shape (N, C_{out}, H_0, W_0) where C_{out} is `out_channels`.

返回类型 Tensor

class mmocr.models.textdet.necks.**FPNF**(*in_channels=[256, 512, 1024, 2048]*, *out_channels=256*, *fusion_type='concat'*, *init_cfg={'distribution': 'uniform', 'layer': 'Conv2d', 'type': 'Xavier'}*)

FPN-like fusion module in Shape Robust Text Detection with Progressive Scale Expansion Network.

参数

- **in_channels** (*list[int]*) –A list of number of input channels.
- **out_channels** (*int*) –The number of output channels.
- **fusion_type** (*str*) –Type of the final feature fusion layer. Available options are "concat" and "add".
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*inputs*)

参数 **inputs** (*list[Tensor]*) –Each tensor has the shape of (N, C_i, H_i, W_i). It usually expects 4 tensors (C2-C5 features) from ResNet.

返回 A tensor of shape (N, C_{out}, H_0, W_0) where C_{out} is `out_channels`.

返回类型 Tensor

class mmocr.models.textdet.necks.**FPN_UNet**(*in_channels*, *out_channels*, *init_cfg={'distribution': 'uniform', 'layer': ['Conv2d', 'ConvTranspose2d'], 'type': 'Xavier'}*)

The class for implementing DRRG and TextSnake U-Net-like FPN.

DRRG: Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection.

TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

参数

- **in_channels** (*list[int]*) –Number of input channels at each scale. The length of the list should be 4.
- **out_channels** (*int*) –The number of output channels.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*x*)

参数 **x** (*list[Tensor] | tuple[Tensor]*) –A list of four tensors of shape (N, C_i, H_i, W_i), representing C2, C3, C4, C5 features respectively. C_i should matches the number in `in_channels`.

返回 Shape (N, C, H, W) where $H = 4H_0$ and $W = 4W_0$.

返回类型 Tensor

24.4 textdet_detectors

class mmocr.models.textdet.detectors.**DBNet**(*backbone*, *neck*, *bbox_head*, *train_cfg=None*, *test_cfg=None*, *pretrained=None*, *show_score=False*, *init_cfg=None*)

The class for implementing DBNet text detector: Real-time Scene Text Detection with Differentiable Binarization.

[https://arxiv.org/abs/1911.08947].

class mmocr.models.textdet.detectors.**DRRG**(*backbone*, *neck*, *bbox_head*, *train_cfg=None*, *test_cfg=None*, *pretrained=None*, *show_score=False*, *init_cfg=None*)

The class for implementing DRRG text detector. Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection.

[https://arxiv.org/abs/2003.07493]

forward_train(*img*, *img_metas*, ***kwargs*)

参数

- **img** (*Tensor*) –Input images of shape (N, C, H, W). Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –A List of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details of the values of these keys see mmdet.datasets.pipelines.Collect.

返回 A dictionary of loss components.

返回类型 dict[str, Tensor]

simple_test(*img*, *img_metas*, *rescale=False*)

Test function without test-time augmentation.

参数

- **img** (*torch.Tensor*) –Images with shape (N, C, H, W).
- **img_metas** (*list[dict]*) –List of image information.
- **rescale** (*bool, optional*) –Whether to rescale the results. Defaults to False.

返回

BBox results of each image and classes. The outer list corresponds to each image. The inner list corresponds to each class.

返回类型 list[list[np.ndarray]]

class mmocr.models.textdet.detectors.**FCENet**(*backbone*, *neck*, *bbox_head*, *train_cfg=None*, *test_cfg=None*, *pretrained=None*, *show_score=False*, *init_cfg=None*)

The class for implementing FCENet text detector FCENet(CVPR2021): Fourier Contour Embedding for Arbitrary-shaped Text

Detection

[https://arxiv.org/abs/2104.10442]

simple_test(*img*, *img_metas*, *rescale=False*)

Test function without test-time augmentation.

参数

- **img** (*torch.Tensor*) –Images with shape (N, C, H, W).
- **img_metas** (*list[dict]*) –List of image information.
- **rescale** (*bool, optional*) –Whether to rescale the results. Defaults to False.

返回

BBox results of each image and classes. The outer list corresponds to each image. The inner list corresponds to each class.

返回类型 list[list[np.ndarray]]

class mmocr.models.textdet.detectors.**OCRMaskRCNN**(*backbone*, *rpn_head*, *roi_head*, *train_cfg*, *test_cfg*, *neck=None*, *pretrained=None*, *text_repr_type='quad'*, *show_score=False*, *init_cfg=None*)

Mask RCNN tailored for OCR.

simple_test(*img*, *img_metas*, *proposals=None*, *rescale=False*)

Test without augmentation.

class mmocr.models.textdet.detectors.**PANet**(*backbone*, *neck*, *bbox_head*, *train_cfg=None*, *test_cfg=None*, *pretrained=None*, *show_score=False*, *init_cfg=None*)

The class for implementing PANet text detector:

Efficient and Accurate Arbitrary-Shaped Text Detection with Pixel Aggregation Network [https://arxiv.org/abs/1908.05900].

class mmocr.models.textdet.detectors.**PSENet**(*backbone*, *neck*, *bbox_head*, *train_cfg=None*, *test_cfg=None*, *pretrained=None*, *show_score=False*, *init_cfg=None*)

The class for implementing PSENet text detector: Shape Robust Text Detection with Progressive Scale Expansion Network.

[https://arxiv.org/abs/1806.02559].

class mmocr.models.textdet.detectors.**SingleStageTextDetector**(*backbone*, *neck*, *bbox_head*, *train_cfg=None*, *test_cfg=None*, *pretrained=None*, *init_cfg=None*)

The class for implementing single stage text detector.

It is the parent class of PANet, PSENet, and DBNet.

forward_train(*img*, *img_metas*, ***kwargs*)

参数

- **img** (*Tensor*) –Input images of shape (N, C, H, W). Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –A list of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details on the values of these keys, see mmdet.datasets.pipelines.Collect.

返回 A dictionary of loss components.

返回类型 dict[str, Tensor]

simple_test(*img*, *img_metas*, *rescale=False*)

Test function without test-time augmentation.

参数

- **img** (*torch.Tensor*) –Images with shape (N, C, H, W).
- **img_metas** (*list[dict]*) –List of image information.
- **rescale** (*bool, optional*) –Whether to rescale the results. Defaults to False.

返回

BBox results of each image and classes. The outer list corresponds to each image. The inner list corresponds to each class.

返回类型 list[list[np.ndarray]]

class mmocr.models.textdet.detectors.**TextDetectorMixin**(*show_score*)

The class for implementing text detector auxiliary methods.

get_boundary(*results*)

Convert segmentation into text boundaries.

参数 **results** (*tuple*) –The result tuple. The first element is segmentation while the second is its scores.

返回 A result dict containing ‘boundary_result’.

返回类型 results (dict)

show_result(*img*, *result*, *score_thr=0.5*, *bbox_color='green'*, *text_color='green'*, *thickness=1*, *font_scale=0.5*, *win_name=''*, *show=False*, *wait_time=0*, *out_file=None*)

Draw *result* over *img*.

参数

- **img** (*str or Tensor*) –The image to be displayed.
- **result** (*dict*) –The results to draw over *img*.
- **score_thr** (*float, optional*) –Minimum score of bboxes to be shown. Default: 0.3.
- **bbox_color** (str or tuple or Color) –Color of bbox lines.
- **text_color** (str or tuple or Color) –Color of texts.
- **thickness** (*int*) –Thickness of lines.
- **font_scale** (*float*) –Font scales of texts.
- **win_name** (*str*) –The window name.
- **wait_time** (*int*) –Value of waitKey param. Default: 0.
- **show** (*bool*) –Whether to show the image. Default: False.
- **out_file** (*str or None*) –The filename to write the image. Default: None.imshow_pred_boundary‘

class mmocr.models.textdet.detectors.**TextSnake**(*backbone*, *neck*, *bbox_head*, *train_cfg=None*, *test_cfg=None*, *pretrained=None*, *show_score=False*, *init_cfg=None*)

The class for implementing TextSnake text detector: TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

[https://arxiv.org/abs/1807.01544]

24.5 textdet_losses

class mmocr.models.textdet.losses.**DBLoss**(*alpha=1*, *beta=1*, *reduction='mean'*, *negative_ratio=3.0*, *eps=1e-06*, *bbce_loss=False*)

The class for implementing DBNet loss.

This is partially adapted from https://github.com/MhLiao/DB.

参数

- **alpha** (*float*) –The binary loss coef.
- **beta** (*float*) –The threshold loss coef.
- **reduction** (*str*) –The way to reduce the loss.
- **negative_ratio** (*float*) –The ratio of positives to negatives.
- **eps** (*float*) –Epsilon in the threshold loss function.
- **bbce_loss** (*bool*) –Whether to use balanced bce for probability loss. If False, dice loss will be used instead.

bitmasks2tensor(*bitmasks*, *target_sz*)

Convert Bitmasks to tensor.

参数

- **bitmasks** (*list[BitmapMasks]*) –The BitmapMasks list. Each item is for one img.
- **target_sz** (*tuple(int, int)*) –The target tensor of size (H, W).

返回 The list of kernel tensors. Each element stands for one kernel level.

返回类型 list[Tensor]

forward(*preds*, *downsample_ratio*, *gt_shrink*, *gt_shrink_mask*, *gt_thr*, *gt_thr_mask*)

Compute DBNet loss.

参数

- **preds** (*Tensor*) –The output tensor with size $(N, 3, H, W)$.
- **downsample_ratio** (*float*) –The downsample ratio for the ground truths.
- **gt_shrink** (*list[BitmapMasks]*) –The mask list with each element being the shrunk text mask for one img.
- **gt_shrink_mask** (*list[BitmapMasks]*) –The effective mask list with each element being the shrunk effective mask for one img.
- **gt_thr** (*list[BitmapMasks]*) –The mask list with each element being the threshold text mask for one img.

- **gt_thr_mask** (*list[BitmapMasks]*) –The effective mask list with each element being the threshold effective mask for one img.

返回 The dict for dbnet losses with "loss_prob", "loss_db" and "loss_thresh".

返回类型 dict

class mmocr.models.textdet.losses.**DRRGLoss**(*ohem_ratio=3.0*)

The class for implementing DRRG loss. This is partially adapted from https://github.com/GXYM/DRRG licensed under the MIT license.

DRRG: Deep Relational Reasoning Graph Network for Arbitrary Shape Text Detection.

参数 **ohem_ratio** (*float*) –The negative/positive ratio in ohem.

balance_bce_loss(*pred*, *gt*, *mask*)

Balanced Binary-CrossEntropy Loss.

参数

- **pred** (*Tensor*) –Shape of $(1, H, W)$.
- **gt** (*Tensor*) –Shape of $(1, H, W)$.
- **mask** (*Tensor*) –Shape of $(1, H, W)$.

返回 Balanced bce loss.

返回类型 Tensor

bitmasks2tensor(*bitmasks*, *target_sz*)

Convert Bitmasks to tensor.

参数

- **bitmasks** (*list[BitmapMasks]*) –The BitmapMasks list. Each item is for one img.
- **target_sz** (*tuple(int, int)*) –The target tensor of size (H, W).

返回 The list of kernel tensors. Each element stands for one kernel level.

返回类型 list[Tensor]

forward(*preds*, *downsample_ratio*, *gt_text_mask*, *gt_center_region_mask*, *gt_mask*, *gt_top_height_map*, *gt_bot_height_map*, *gt_sin_map*, *gt_cos_map*)

Compute Drrg loss.

参数

- **preds** (*tuple(Tensor)*) –The first is the prediction map with shape (N, C_{out}, H, W). The second is prediction from GCN module, with shape $(N, 2)$. The third is ground-truth label with shape $(N, 8)$.
- **downsample_ratio** (*float*) –The downsample ratio.
- **gt_text_mask** (*list[BitmapMasks]*) –Text mask.

- **gt_center_region_mask** (*list[BitmapMasks]*) –Center region mask.
- **gt_mask** (*list[BitmapMasks]*) –Effective mask.
- **gt_top_height_map** (*list[BitmapMasks]*) –Top height map.
- **gt_bot_height_map** (*list[BitmapMasks]*) –Bottom height map.
- **gt_sin_map** (*list[BitmapMasks]*) –Sinusoid map.
- **gt_cos_map** (*list[BitmapMasks]*) –Cosine map.

返回 A loss dict with `loss_text`, `loss_center`, `loss_height`, `loss_sin`, `loss_cos`, and `loss_gcn`.

返回类型 dict

gcn_loss(*gcn_data*)

CrossEntropy Loss from gcn module.

参数 **gcn_data** (*tuple(Tensor, Tensor)*) –The first is the prediction with shape $(N, 2)$ and the second is the gt label with shape (m, n) where $m * n = N$.

返回 CrossEntropy loss.

返回类型 Tensor

class mmocr.models.textdet.losses.**FCELoss**(*fourier_degree*, *num_sample*, *ohem_ratio=3.0*)

The class for implementing FCENet loss.

FCENet(CVPR2021): Fourier Contour Embedding for Arbitrary-shaped Text Detection

参数

- **fourier_degree** (*int*) –The maximum Fourier transform degree k.
- **num_sample** (*int*) –The sampling points number of regression loss. If it is too small, fcenet tends to be overfitting.
- **ohem_ratio** (*float*) –the negative/positive ratio in OHEM.

forward(*preds*, *_*, *p3_maps*, *p4_maps*, *p5_maps*)

Compute FCENet loss.

参数

- **preds** (*list[list[Tensor]]*) –The outer list indicates images in a batch, and the inner list indicates the classification prediction map (with shape (N, C, H, W)) and regression map (with shape (N, C, H, W)).
- **p3_maps** (*list[ndarray]*) –List of leval 3 ground truth target map with shape (C, H, W).
- **p4_maps** (*list[ndarray]*) –List of leval 4 ground truth target map with shape (C, H, W).

- **p5_maps** (*list[ndarray]*) –List of leval 5 ground truth target map with shape (C, H, W).

返回 A loss dict with `loss_text`, `loss_center`, `loss_reg_x` and `loss_reg_y`.

返回类型 dict

fourier2poly(*real_maps*, *imag_maps*)

Transform Fourier coefficient maps to polygon maps.

参数

- **real_maps** (*tensor*) –A map composed of the real parts of the Fourier coefficients, whose shape is (-1, 2k+1)
- **imag_maps** (*tensor*) –A map composed of the imag parts of the Fourier coefficients, whose shape is (-1, 2k+1)

Returns

x_maps (tensor): A map composed of the x value of the polygon represented by n sample points (xn, yn), whose shape is (-1, n)

y_maps (tensor): A map composed of the y value of the polygon represented by n sample points (xn, yn), whose shape is (-1, n)

class mmocr.models.textdet.losses.**PANLoss**(*alpha=0.5*, *beta=0.25*, *delta_aggregation=0.5*, *delta_discrimination=3*, *ohem_ratio=3*, *reduction='mean'*, *speedup_bbox_thr=-1*)

The class for implementing PANet loss. This was partially adapted from https://github.com/WenmuZhou/PAN.pytorch.

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

参数

- **alpha** (*float*) –The kernel loss coef.
- **beta** (*float*) –The aggregation and discriminative loss coef.
- **delta_aggregation** (*float*) –The constant for aggregation loss.
- **delta_discrimination** (*float*) –The constant for discriminative loss.
- **ohem_ratio** (*float*) –The negative/positive ratio in ohem.
- **reduction** (*str*) –The way to reduce the loss.
- **speedup_bbox_thr** (*int*) –Speed up if speedup_bbox_thr > 0 and < bbox num.

aggregation_discrimination_loss(*gt_texts*, *gt_kernels*, *inst_embeds*)

Compute the aggregation and discrimnative losses.

参数

- **gt_texts** (*Tensor*) –The ground truth text mask of size $(N, 1, H, W)$.
- **gt_kernels** (*Tensor*) –The ground truth text kernel mask of size $(N, 1, H, W)$.
- **inst_embeds** (*Tensor*) –The text instance embedding tensor of size $(N, 1, H, W)$.

返回 A tuple of aggregation loss and discriminative loss before reduction.

返回类型 (Tensor, Tensor)

bitmasks2tensor(*bitmasks*, *target_sz*)

Convert Bitmasks to tensor.

参数

- **bitmasks** (*list[BitmapMasks]*) –The BitmapMasks list. Each item is for one img.
- **target_sz** (*tuple(int, int)*) –The target tensor of size (H, W).

返回 The list of kernel tensors. Each element stands for one kernel level.

返回类型 list[Tensor]

forward(*preds*, *downsample_ratio*, *gt_kernels*, *gt_mask*)

Compute PANet loss.

参数

- **preds** (*Tensor*) –The output tensor of size $(N, 6, H, W)$.
- **downsample_ratio** (*float*) –The downsample ratio between preds and the input img.
- **gt_kernels** (*list[BitmapMasks]*) –The kernel list with each element being the text kernel mask for one img.
- **gt_mask** (*list[BitmapMasks]*) –The effective mask list with each element being the effective mask for one img.

返回 A loss dict with `loss_text`, `loss_kernel`, `loss_aggregation` and `loss_discrimination`.

返回类型 dict

ohem_batch(*text_scores*, *gt_texts*, *gt_mask*)

OHEM sampling for a batch of imgs.

参数

- **text_scores** (*Tensor*) –The text scores of size (H, W).
- **gt_texts** (*Tensor*) –The gt text masks of size (H, W).
- **gt_mask** (*Tensor*) –The gt effective mask of size (H, W).

返回 The sampled mask of size (H, W).

返回类型 Tensor

ohem_img(*text_score*, *gt_text*, *gt_mask*)

Sample the top-k maximal negative samples and all positive samples.

参数

- **text_score** (*Tensor*) –The text score of size (H, W).
- **gt_text** (*Tensor*) –The ground truth text mask of size (H, W).
- **gt_mask** (*Tensor*) –The effective region mask of size (H, W).

返回 The sampled pixel mask of size (H, W).

返回类型 Tensor

class mmocr.models.textdet.losses.**PSELoss**(*alpha=0.7*, *ohem_ratio=3*, *reduction='mean'*, *kernel_sample_type='adaptive'*)

The class for implementing PSENet loss. This is partially adapted from https://github.com/whai362/PSENet.

PSENet: Shape Robust Text Detection with Progressive Scale Expansion Network.

参数

- **alpha** (*float*) –Text loss coefficient, and $1 - \alpha$ is the kernel loss coefficient.
- **ohem_ratio** (*float*) –The negative/positive ratio in ohem.
- **reduction** (*str*) –The way to reduce the loss. Available options are "mean" and "sum" .

forward(*score_maps*, *downsample_ratio*, *gt_kernels*, *gt_mask*)

Compute PSENet loss.

参数

- **score_maps** (*tensor*) –The output tensor with size of Nx6xHxW.
- **downsample_ratio** (*float*) –The downsample ratio between score_maps and the input img.
- **gt_kernels** (*list[BitmapMasks]*) –The kernel list with each element being the text kernel mask for one img.
- **gt_mask** (*list[BitmapMasks]*) –The effective mask list with each element being the effective mask for one img.

返回 A loss dict with `loss_text` and `loss_kernel`.

返回类型 dict

class mmocr.models.textdet.losses.**TextSnakeLoss**(*ohem_ratio=3.0*)

The class for implementing TextSnake loss. This is partially adapted from https://github.com/princewang1994/TextSnake.pytorch.

TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

参数 **ohem_ratio** (*float*) –The negative/positive ratio in ohem.

bitmasks2tensor (*bitmasks*, *target_sz*)

Convert Bitmasks to tensor.

参数

- **bitmasks** (*list[BitmapMasks]*) –The BitmapMasks list. Each item is for one img.
- **target_sz** (*tuple(int, int)*) –The target tensor of size (H, W).

返回 The list of kernel tensors. Each element stands for one kernel level.

返回类型 list[Tensor]

forward (*pred_maps*, *downsample_ratio*, *gt_text_mask*, *gt_center_region_mask*, *gt_mask*, *gt_radius_map*, *gt_sin_map*, *gt_cos_map*)

参数

- **pred_maps** (*Tensor*) –The prediction map of shape $(N, 5, H, W)$, where each dimension is the map of "text_region", "center_region", "sin_map", "cos_map", and "radius_map" respectively.
- **downsample_ratio** (*float*) –Downsample ratio.
- **gt_text_mask** (*list[BitmapMasks]*) –Gold text masks.
- **gt_center_region_mask** (*list[BitmapMasks]*) –Gold center region masks.
- **gt_mask** (*list[BitmapMasks]*) –Gold general masks.
- **gt_radius_map** (*list[BitmapMasks]*) –Gold radius maps.
- **gt_sin_map** (*list[BitmapMasks]*) –Gold sin maps.
- **gt_cos_map** (*list[BitmapMasks]*) –Gold cos maps.

返回 A loss dict with `loss_text`, `loss_center`, `loss_radius`, `loss_sin` and `loss_cos`.

返回类型 dict

24.6 textdet_postprocess

24.7 textrecog_recognizer

class mmocr.models.textrecog.recognizer.**BaseRecognizer** (*init_cfg=None*)

Base class for text recognition.

abstract aug_test(*imgs*, *img_metas*, ***kwargs*)

Test function with test time augmentation.

参数

- **imgs** (*list[tensor]*) –Tensor should have shape NxCxHxW, which contains all images in the batch.
- **img_metas** (*list[list[dict]]*) –The metadata of images.

abstract extract_feat(*imgs*)

Extract features from images.

forward(*img*, *img_metas*, *return_loss=True*, ***kwargs*)

Calls either *forward_train()* or *forward_test()* depending on whether `return_loss` is `True`.

Note that img and img_meta are single-nested (i.e. tensor and list[dict]).

forward_test(*imgs*, *img_metas*, ***kwargs*)

参数

- **imgs** (*tensor | list[tensor]*) –Tensor should have shape NxCxHxW, which contains all images in the batch.
- **img_metas** (*list[dict] | list[list[dict]]*) –The outer list indicates images in a batch.

abstract forward_train(*imgs*, *img_metas*, ***kwargs*)

参数

- **img** (*tensor*) –tensors with shape (N, C, H, W). Typically should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –List of image info dict where each dict has: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details of the values of these keys, see `mmdet.datasets.pipelines.Collect`.
- **kwargs** (*keyword arguments*) –Specific to concrete implementation.

show_result(*img*, *result*, *gt_label=''*, *win_name=''*, *show=False*, *wait_time=0*, *out_file=None*, ***kwargs*)

Draw *result* on *img*.

参数

- **img** (*str or tensor*) –The image to be displayed.
- **result** (*dict*) –The results to draw on *img*.
- **gt_label** (*str*) –Ground truth label of img.

- **win_name** (*str*) –The window name.
- **wait_time** (*int*) –Value of waitKey param. Default: 0.
- **show** (*bool*) –Whether to show the image. Default: False.
- **out_file** (*str or None*) –The output filename. Default: None.

返回 Only if not *show* or *out_file*.

返回类型 img (tensor)

train_step(*data*, *optimizer*)

The iteration step during training.

This method defines an iteration step during training, except for the back propagation and optimizer update, which are done by an optimizer hook. Note that in some complicated cases or models (e.g. GAN), the whole process (including the back propagation and optimizer update) is also defined by this method.

参数

- **data** (*dict*) –The outputs of dataloader.
- **optimizer** (`torch.optim.Optimizer` | dict) –The optimizer of runner is passed to `train_step()`. This argument is unused and reserved.

返回

It should contain at least 3 keys: loss, log_vars, `num_samples`.

- `loss` is a tensor for back propagation, which is a

weighted sum of multiple losses. - `log_vars` contains all the variables to be sent to the logger. - `num_samples` indicates the batch size used for averaging the logs (Note: for the DDP model, num_samples refers to the batch size for each GPU).

返回类型 dict

val_step(*data*, *optimizer*)

The iteration step during validation.

This method shares the same signature as *train_step()*, but is used during val epochs. Note that the evaluation after training epochs is not implemented by this method, but by an evaluation hook.

class mmocr.models.textrecog.recognizer.**CRNNNet**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *loss=None*, *label_convertor=None*, *train_cfg=None*, *test_cfg=None*, *max_seq_len=40*, *pretrained=None*, *init_cfg=None*)

CTC-loss based recognizer.

class mmocr.models.textrecog.recognizer.**EncodeDecodeRecognizer**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *loss=None*, *label_convertor=None*, *train_cfg=None*, *test_cfg=None*, *max_seq_len=40*, *pretrained=None*, *init_cfg=None*)

Base class for encode-decode recognizer.

aug_test(*imgs*, *img_metas*, ***kwargs*)

Test function as well as time augmentation.

参数

- **imgs** (*list[tensor]*) –Tensor should have shape NxCxHxW, which contains all images in the batch.
- **img_metas** (*list[list[dict]]*) –The metadata of images.

extract_feat(*img*)

Directly extract features from the backbone.

forward_train(*img*, *img_metas*)

参数

- **img** (*tensor*) –Input images of shape (N, C, H, W). Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –A list of image info dict where each dict contains: 'img_shape', 'filename', and may also contain 'ori_shape', and 'img_norm_cfg'. For details on the values of these keys see mmdet.datasets.pipelines.Collect.

返回 A dictionary of loss components.

返回类型 dict[str, tensor]

simple_test(*img*, *img_metas*, ***kwargs*)

Test function with test time augmentation.

参数

- **imgs** (*torch.Tensor*) –Image input tensor.
- **img_metas** (*list[dict]*) –List of image information.

返回 Text label result of each image.

返回类型 list[str]

class mmocr.models.textrecog.recognizer.**NRTR**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *loss=None*, *label_convertor=None*, *train_cfg=None*, *test_cfg=None*, *max_seq_len=40*, *pretrained=None*, *init_cfg=None*)

Implementation of NRTR

class mmocr.models.textrecog.recognizer.**RobustScanner**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *loss=None*, *label_convertor=None*, *train_cfg=None*, *test_cfg=None*, *max_seq_len=40*, *pretrained=None*, *init_cfg=None*)

Implementation of 'RobustScanner.

<https://arxiv.org/pdf/2007.07542.pdf>

class mmocr.models.textrecog.recognizer.**SARNet**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *loss=None*, *label_convertor=None*, *train_cfg=None*, *test_cfg=None*, *max_seq_len=40*, *pretrained=None*, *init_cfg=None*)

Implementation of SAR

class mmocr.models.textrecog.recognizer.**SATRN**(*preprocessor=None*, *backbone=None*, *encoder=None*, *decoder=None*, *loss=None*, *label_convertor=None*, *train_cfg=None*, *test_cfg=None*, *max_seq_len=40*, *pretrained=None*, *init_cfg=None*)

Implementation of SATRN

class mmocr.models.textrecog.recognizer.**SegRecognizer**(*preprocessor=None*, *backbone=None*, *neck=None*, *head=None*, *loss=None*, *label_convertor=None*, *train_cfg=None*, *test_cfg=None*, *pretrained=None*, *init_cfg=None*)

Base class for segmentation based recognizer.

aug_test(*imgs*, *img_metas*, ***kwargs*)

Test function with test time augmentation.

参数

- **imgs** (*list[tensor]*) –Tensor should have shape NxCxHxW, which contains all images in the batch.
- **img_metas** (*list[list[dict]]*) –The metadata of images.

extract_feat(*img*)

Directly extract features from the backbone.

forward_train(*img*, *img_metas*, *gt_kernels=None*)

参数

- **img** (*tensor*) –Input images of shape (N, C, H, W). Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –A list of image info dict where each dict contains: 'img_shape', 'filename', and may also contain 'ori_shape', and 'img_norm_cfg'. For details on the values of these keys see mmdet.datasets.pipelines.Collect.

返回 A dictionary of loss components.

返回类型 dict[str, tensor]

simple_test(*img*, *img_metas*, ***kwargs*)

Test function without test time augmentation.

参数

- **imgs** (*torch.Tensor*) –Image input tensor.
- **img_metas** (*list[dict]*) –List of image information.

返回 Text label result of each image.

返回类型 list[str]

24.8 textrecog_backbones

class mmocr.models.textrecog.backbones.**NRTRModalityTransform**(*input_channels=3*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.backbones.**ResNet31OCR**(*base_channels=3*, *layers=[1, 2, 5, 3]*, *channels=[64, 128, 256, 256, 512, 512, 512]*, *out_indices=None*, *stage4_pool_cfg={'kernel_size': (2, 1), 'stride': (2, 1)}*, *last_stage_pool=False*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

Implement ResNet backbone for text recognition, modified from ResNet

参数

- **base_channels** (*int*) –Number of channels of input image tensor.
- **layers** (*list[int]*) –List of BasicBlock number for each stage.
- **channels** (*list[int]*) –List of out_channels of Conv2d layer.
- **out_indices** (*None | Sequence[int]*) –Indices of output stages.
- **stage4_pool_cfg** (*dict*) –Dictionary to construct and configure pooling layer in stage 4.
- **last_stage_pool** (*bool*) –If True, add *MaxPool2d* layer to last stage.

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.backbones.**ShallowCNN**(*input_channels=1*, *hidden_dim=512*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

Implement Shallow CNN block for SATRN.

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

参数

- **base_channels** (*int*) –Number of channels of input image tensor D_i.
- **hidden_dim** (*int*) –Size of hidden layers of the model D_m.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*x*)

参数 **x** (*Tensor*) –Input image feature (N, D_i, H, W).

返回 A tensor of shape $(N, D_m, H/4, W/4)$.

返回类型 Tensor

class mmocr.models.textrecog.backbones.**VeryDeepVgg**(*leaky_relu=True*, *input_channels=3*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

Implement VGG-VeryDeep backbone for text recognition, modified from VGG-VeryDeep

参数

- **leaky_relu** (*bool*) –Use leakyRelu or not.
- **input_channels** (*int*) –Number of channels of input image tensor.

forward(*x*)

参数 **x** (*Tensor*) –Images of shape (N, C, H, W).

返回 The feature Tensor of shape $(N, 512, H/32, (W/4 + 1))$.

返回类型 Tensor

24.9 textrecog_necks

class mmocr.models.textrecog.necks.**FPNOCR**(*in_channels*, *out_channels*, *last_stage_only=True*, *init_cfg=None*)

FPN-like Network for segmentation based text recognition.

参数

- **in_channels** (*list[int]*) –Number of input channels C_i for each scale.
- **out_channels** (*int*) –Number of output channels C_{out} for each scale.
- **last_stage_only** (*bool*) –If True, output last stage only.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*inputs*)

参数 **inputs** (*list[Tensor]*) –A list of n tensors. Each tensor has the shape of (N, C_i, H_i, W_i). It usually expects 4 tensors (C2-C5 features) from ResNet.

返回 A tuple of n-1 tensors. Each has the of shape $(N, C_{out}, H_{n-2-i}, W_{n-2-i})$. If `last_stage_only=True` (default), the size of the tuple is 1 and only the last element will be returned.

返回类型 tuple(Tensor)

24.10 textrecog_heads

class mmocr.models.textrecog.heads.**SegHead**(*in_channels=128*, *num_classes=37*, *upsample_param=None*, *init_cfg=None*)

Head for segmentation based text recognition.

参数

- **in_channels** (*int*) –Number of input channels C.
- **num_classes** (*int*) –Number of output classes C_{out}.
- **upsample_param** (*dict | None*) –Config dict for interpolation layer. Default: `dict(scale_factor=1.0, mode='nearest')`
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*out_neck*)

参数 **out_neck** (*list[Tensor]*) –A list of tensor of shape (N, C_i, H_i, W_i). The network only uses the last one (`out_neck[-1]`).

返回 A tensor of shape (N, C_{out}, kH, kW) where k is determined by `upsample_param`.

返回类型 Tensor

24.11 textrecog_convertors

class mmocr.models.textrecog.convertors.**AttnConvertor**(*dict_type='DICT90'*, *dict_file=None*, *dict_list=None*, *with_unknown=True*, *max_seq_len=40*, *lower=False*, *start_end_same=True*, ***kwargs*)

Convert between text, index and tensor for encoder-decoder based pipeline.

参数

- **dict_type** (*str*) –Type of dict, should be one of {‘DICT36’, ‘DICT90’}.
- **dict_file** (*None|str*) –Character dict file path. If not none, higher priority than dict_type.
- **dict_list** (*None|list[str]*) –Character list. If not none, higher priority than dict_type, but lower than dict_file.
- **with_unknown** (*bool*) –If True, add *UKN* token to class.
- **max_seq_len** (*int*) –Maximum sequence length of label.
- **lower** (*bool*) –If True, convert original string to lower case.
- **start_end_same** (*bool*) –Whether use the same index for start and end token or not. Default: True.

str2tensor(*strings*)

Convert text-string into tensor. :param strings: [‘hello’, ‘world’] :type strings: list[str]

返回

Tensor | list[tensor]):

tensors (list[Tensor]): [torch.Tensor([1,2,3,3,4]), torch.Tensor([5,4,6,3,7])]

padded_targets (Tensor(bsz * max_seq_len))

返回类型 dict (str

tensor2idx(*outputs*, *img_metas=None*)

Convert output tensor to text-index :param outputs: model outputs with size: N * T * C :type outputs: tensor :param img_metas: Each dict contains one image info. :type img_metas: list[dict]

返回

[[1,2,3,3,4], [5,4,6,3,7]] scores (list[list[float]]): [[0.9,0.8,0.95,0.97,0.94],

[0.9,0.9,0.98,0.97,0.96]]

返回类型 indexes (list[list[int]])

class mmocr.models.textrecog.convertors.**BaseConvertor**(*dict_type='DICT90'*, *dict_file=None*, *dict_list=None*)

Convert between text, index and tensor for text recognize pipeline.

参数

- **dict_type** (*str*) –Type of dict, should be either ‘DICT36’ or ‘DICT90’.
- **dict_file** (*None|str*) –Character dict file path. If not none, the dict_file is of higher priority than dict_type.
- **dict_list** (*None|list[str]*) –Character list. If not none, the list is of higher priority than dict_type, but lower than dict_file.

idx2str(*indexes*)

Convert indexes to text strings.

参数 **indexes** (*list[list[int]]*) –[[1,2,3,3,4], [5,4,6,3,7]].

返回 [‘hello’, ‘world’].

返回类型 strings (list[str])

num_classes()

Number of output classes.

str2idx(*strings*)

Convert strings to indexes.

参数 **strings** (*list[str]*) –[‘hello’, ‘world’].

返回 [[1,2,3,3,4], [5,4,6,3,7]].

返回类型 indexes (list[list[int]])

str2tensor(*strings*)

Convert text-string to input tensor.

参数 **strings** (*list[str]*) –[‘hello’, ‘world’].

返回

[torch.Tensor([1,2,3,3,4]), torch.Tensor([5,4,6,3,7])].

返回类型 tensors (list[torch.Tensor])

tensor2idx(*output*)

Convert model output tensor to character indexes and scores. :param output: The model outputs with size: N * T * C :type output: tensor

返回

[[1,2,3,3,4], [5,4,6,3,7]]. scores (list[list[float]]): [[0.9,0.8,0.95,0.97,0.94],

[0.9,0.9,0.98,0.97,0.96]].

返回类型 indexes (list[list[int]])

class mmocr.models.textrecog.convertors.**CTCConvertor**(*dict_type='DICT90'*, *dict_file=None*, *dict_list=None*, *with_unknown=True*, *lower=False*, ***kwargs*)

Convert between text, index and tensor for CTC loss-based pipeline.

参数

- **dict_type** (*str*) –Type of dict, should be either 'DICT36' or 'DICT90'.
- **dict_file** (*None | str*) –Character dict file path. If not none, the file is of higher priority than dict_type.
- **dict_list** (*None | list[str]*) –Character list. If not none, the list is of higher priority than dict_type, but lower than dict_file.
- **with_unknown** (*bool*) –If True, add *UKN* token to class.
- **lower** (*bool*) –If True, convert original string to lower case.

str2tensor(*strings*)

Convert text-string to ctc-loss input tensor.

参数 strings (*list[str]*) –['hello', 'world'].

返回

tensor | list[tensor]):

tensors (list[tensor]): [torch.Tensor([1,2,3,3,4]), torch.Tensor([5,4,6,3,7])].

flatten_targets (tensor): torch.Tensor([1,2,3,3,4,5,4,6,3,7]). target_lengths (tensor): torch.IntTensot([5,5]).

返回类型 dict (str

tensor2idx(*output*, *img_metas*, *topk=1*, *return_topk=False*)

Convert model output tensor to index-list. :param output: The model outputs with size: N * T * C. :type output: tensor :param img_metas: Each dict contains one image info. :type img_metas: list[dict] :param topk: The highest k classes to be returned. :type topk: int :param return_topk: Whether to return topk or just top1. :type return_topk: bool

返回

[[1,2,3,3,4], [5,4,6,3,7]]. scores (list[list[float]]): [[0.9,0.8,0.95,0.97,0.94],

[0.9,0.9,0.98,0.97,0.96]] (

indexes_topk (list[list[list[int]->len=topk]]): scores_topk (list[list[list[float]->len=topk]])

).

返回类型 indexes (list[list[int]])

class mmocr.models.textrecog.convertors.**SegConvertor**(*dict_type='DICT36'*, *dict_file=None*, *dict_list=None*, *with_unknown=True*, *lower=False*, ***kwargs*)

Convert between text, index and tensor for segmentation based pipeline.

参数

- **dict_type** (*str*) –Type of dict, should be either 'DICT36' or 'DICT90'.
- **dict_file** (*None | str*) –Character dict file path. If not none, the file is of higher priority than dict_type.
- **dict_list** (*None | list[str]*) –Character list. If not none, the list
- **of higher priority than dict_type** (*is*) –
- **lower than dict_file.** (*but*) –
- **with_unknown** (*bool*) –If True, add *UKN* token to class.
- **lower** (*bool*) –If True, convert original string to lower case.

tensor2str(*output*, *img_metas=None*)

Convert model output tensor to string labels. :param output: Model outputs with size: N * C * H * W :type output: tensor :param img_metas: Each dict contains one image info. :type img_metas: list[dict]

返回 Decoded text labels. scores (list[list[float]]): Decoded chars scores.

返回类型 texts (list[str])

24.12 textrecog_encoders

class mmocr.models.textrecog.encoders.**BaseEncoder**(*init_cfg=None*)

Base Encoder class for text recognition.

forward(*feat*, ***kwargs*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while

the latter silently ignores them.

---

class mmocr.models.textrecog.encoders.**ChannelReductionEncoder**(*in_channels*, *out_channels*, *init_cfg={'layer': 'Conv2d', 'type': 'Xavier'}*)

Change the channel number with a one by one convoluational layer.

参数

- **in_channels** (*int*) –Number of input channels.
- **out_channels** (*int*) –Number of output channels.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*feat*, *img_metas=None*)

参数

- **feat** (*Tensor*) –Image features with the shape of (N, C_{in}, H, W).
- **img_metas** (*None*) –Unused.

返回 A tensor of shape (N, C_{out}, H, W).

返回类型 Tensor

class mmocr.models.textrecog.encoders.**NRTREncoder**(*n_layers=6*, *n_head=8*, *d_k=64*, *d_v=64*, *d_model=512*, *d_inner=256*, *dropout=0.1*, *init_cfg=None*, ***kwargs*)

Transformer Encoder block with self attention mechanism.

参数

- **n_layers** (*int*) –The number of sub-encoder-layers in the encoder (default=6).
- **n_head** (*int*) –The number of heads in the multiheadattention models (default=8).
- **d_k** (*int*) –Total number of features in key.
- **d_v** (*int*) –Total number of features in value.
- **d_model** (*int*) –The number of expected features in the decoder inputs (default=512).
- **d_inner** (*int*) –The dimension of the feedforward network model (default=256).
- **dropout** (*float*) –Dropout layer on attn_output_weights.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*feat*, *img_metas=None*)

参数

- **feat** (*Tensor*) –Backbone output of shape (N, C, H, W).
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 The encoder output tensor. Shape (N, T, C).

返回类型 Tensor

class mmocr.models.textrecog.encoders.**SAREncoder**(*enc_bi_rnn=False*, *enc_do_rnn=0.0*, *enc_gru=False*, *d_model=512*, *d_enc=512*, *mask=True*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*, ***kwargs*)

Implementation of encoder module in ‘SAR.

<https://arxiv.org/abs/1811.00751>‘_.

参数

- **enc_bi_rnn** (*bool*) –If True, use bidirectional RNN in encoder.
- **enc_do_rnn** (*float*) –Dropout probability of RNN layer in encoder.
- **enc_gru** (*bool*) –If True, use GRU, else LSTM in encoder.
- **d_model** (*int*) –Dim D_i of channels from backbone.
- **d_enc** (*int*) –Dim D_m of encoder RNN layer.
- **mask** (*bool*) –If True, mask padding in RNN sequence.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*feat*, *img_metas=None*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A tensor of shape (N, D_m).

返回类型 Tensor

class mmocr.models.textrecog.encoders.**SatrnEncoder**(*n_layers=12*, *n_head=8*, *d_k=64*, *d_v=64*, *d_model=512*, *n_position=100*, *d_inner=256*, *dropout=0.1*, *init_cfg=None*, ***kwargs*)

Implement encoder for SATRN, see 'SATRN.

<https://arxiv.org/abs/1910.04396>'_.

参数

- **n_layers** (*int*) –Number of attention layers.
- **n_head** (*int*) –Number of parallel attention heads.
- **d_k** (*int*) –Dimension of the key vector.
- **d_v** (*int*) –Dimension of the value vector.
- **d_model** (*int*) –Dimension D_m of the input from previous model.
- **n_position** (*int*) –Length of the positional encoding vector. Must be greater than `max_seq_len`.
- **d_inner** (*int*) –Hidden dimension of feedforward layers.
- **dropout** (*float*) –Dropout rate.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*feat*, *img_metas=None*)

参数

- **feat** (*Tensor*) –Feature tensor of shape (N, D_m, H, W).
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A tensor of shape (N, T, D_m).

返回类型 Tensor

24.13 textrecog_decoders

class mmocr.models.textrecog.decoders.**BaseDecoder**(*init_cfg=None*, ***kwargs*)

Base decoder class for text recognition.

forward(*feat*, *out_enc*, *targets_dict=None*, *img_metas=None*, *train_mode=True*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

class mmocr.models.textrecog.decoders.**CRNNDecoder**(*in_channels=None*, *num_classes=None*, *rnn_flag=False*, *init_cfg={'layer': 'Conv2d', 'type': 'Xavier'}*, ***kwargs*)

Decoder for CRNN.

参数

- **in_channels** (*int*) –Number of input channels.
- **num_classes** (*int*) –Number of output classes.
- **rnn_flag** (*bool*) –Use RNN or CNN as the decoder.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward_test(*feat*, *out_enc*, *img_metas*)

参数 **feat** (*Tensor*) –A Tensor of shape $(N, H, 1, W)$.

返回 The raw logit tensor. Shape (N, W, C) where C is `num_classes`.

返回类型 Tensor

forward_train(*feat*, *out_enc*, *targets_dict*, *img_metas*)

参数 **feat** (*Tensor*) –A Tensor of shape $(N, H, 1, W)$.

返回 The raw logit tensor. Shape (N, W, C) where C is `num_classes`.

返回类型 Tensor

class mmocr.models.textrecog.decoders.**NRTRDecoder**(*n_layers=6*, *d_embedding=512*, *n_head=8*, *d_k=64*, *d_v=64*, *d_model=512*, *d_inner=256*, *n_position=200*, *dropout=0.1*, *num_classes=93*, *max_seq_len=40*, *start_idx=1*, *padding_idx=92*, *init_cfg=None*, ***kwargs*)

Transformer Decoder block with self attention mechanism.

参数

- **n_layers** (*int*) –Number of attention layers.
- **d_embedding** (*int*) –Language embedding dimension.

- **n_head** (*int*) –Number of parallel attention heads.
- **d_k** (*int*) –Dimension of the key vector.
- **d_v** (*int*) –Dimension of the value vector.
- **d_model** (*int*) –Dimension D_m of the input from previous model.
- **d_inner** (*int*) –Hidden dimension of feedforward layers.
- **n_position** (*int*) –Length of the positional encoding vector. Must be greater than `max_seq_len`.
- **dropout** (*float*) –Dropout rate.
- **num_classes** (*int*) –Number of output classes C.
- **max_seq_len** (*int*) –Maximum output sequence length T.
- **start_idx** (*int*) –The index of *<SOS>*.
- **padding_idx** (*int*) –The index of *<PAD>*.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

警告: This decoder will not predict the final class which is assumed to be *<PAD>*. Therefore, its output size is always $C - 1$. *<PAD>* is also ignored by loss as specified in *mmocr.models.textrecog.recognizer.EncodeDecodeRecognizer*.

forward_train(*feat*, *out_enc*, *targets_dict*, *img_metas*)

参数

- **feat** (*None*) –Unused.
- **out_enc** (*Tensor*) –Encoder output of shape (N, T, D_m) where D_m is `d_model`.
- **targets_dict** (*dict*) –A dict with the key `padded_targets`, a tensor of shape (N, T). Each element is the index of a character.
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 The raw logit tensor. Shape (N, T, C).

返回类型 Tensor

static get_subsequent_mask(*seq*)

For masking out the subsequent info.

class mmocr.models.textrecog.decoders.**ParallelSARDecoder**(*num_classes=37*, *enc_bi_rnn=False*, *dec_bi_rnn=False*, *dec_do_rnn=0.0*, *dec_gru=False*, *d_model=512*, *d_enc=512*, *d_k=64*, *pred_dropout=0.0*, *max_seq_len=40*, *mask=True*, *start_idx=0*, *padding_idx=92*, *pred_concat=False*, *init_cfg=None*, ***kwargs*)

Implementation Parallel Decoder module in ‘SAR.

<https://arxiv.org/abs/1811.00751>‘_.

参数

- **num_classes** (*int*) –Output class number C.
- **channels** (*list[int]*) –Network layer channels.
- **enc_bi_rnn** (*bool*) –If True, use bidirectional RNN in encoder.
- **dec_bi_rnn** (*bool*) –If True, use bidirectional RNN in decoder.
- **dec_do_rnn** (*float*) –Dropout of RNN layer in decoder.
- **dec_gru** (*bool*) –If True, use GRU, else LSTM in decoder.
- **d_model** (*int*) –Dim of channels from backbone D_i.
- **d_enc** (*int*) –Dim of encoder RNN layer D_m.
- **d_k** (*int*) –Dim of channels of attention module.
- **pred_dropout** (*float*) –Dropout probability of prediction layer.
- **max_seq_len** (*int*) –Maximum sequence length for decoding.
- **mask** (*bool*) –If True, mask padding in feature map.
- **start_idx** (*int*) –Index of start token.
- **padding_idx** (*int*) –Index of padding token.
- **pred_concat** (*bool*) –If True, concat glimpse feature from attention with holistic feature and hidden state.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

警告: This decoder will not predict the final class which is assumed to be <*PAD*>. Therefore, its output size is always $C - 1$. <*PAD*> is also ignored by loss as specified in *mmocr.models.textrecog.recognizer.EncodeDecodeRecognizer*.

forward_test (*feat*, *out_enc*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A raw logit tensor of shape $(N, T, C - 1)$.

返回类型 Tensor

forward_train (*feat*, *out_enc*, *targets_dict*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **targets_dict** (*dict*) –A dict with the key `padded_targets`, a tensor of shape (N, T). Each element is the index of a character.
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A raw logit tensor of shape $(N, T, C - 1)$.

返回类型 Tensor

class mmocr.models.textrecog.decoders.**ParallelSARDecoderWithBS**(*beam_width=5*, *num_classes=37*, *enc_bi_rnn=False*, *dec_bi_rnn=False*, *dec_do_rnn=0*, *dec_gru=False*, *d_model=512*, *d_enc=512*, *d_k=64*, *pred_dropout=0.0*, *max_seq_len=40*, *mask=True*, *start_idx=0*, *padding_idx=0*, *pred_concat=False*, *init_cfg=None*, ***kwargs*)

Parallel Decoder module with beam-search in SAR.

参数 **beam_width** (*int*) –Width for beam search.

forward_test(*feat*, *out_enc*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A raw logit tensor of shape $(N, T, C-1)$.

返回类型 Tensor

class mmocr.models.textrecog.decoders.**PositionAttentionDecoder**(*num_classes=None*, *rnn_layers=2*, *dim_input=512*, *dim_model=128*, *max_seq_len=40*, *mask=True*, *return_feature=False*, *encode_value=False*, *init_cfg=None*)

Position attention decoder for RobustScanner.

RobustScanner: RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

参数

- **num_classes** (*int*) –Number of output classes C.
- **rnn_layers** (*int*) –Number of RNN layers.
- **dim_input** (*int*) –Dimension D_i of input vector `feat`.
- **dim_model** (*int*) –Dimension D_m of the model. Should also be the same as encoder output vector `out_enc`.
- **max_seq_len** (*int*) –Maximum output sequence length T.
- **mask** (*bool*) –Whether to mask input features according to `img_meta['valid_ratio']`.
- **return_feature** (*bool*) –Return feature or logits as the result.
- **encode_value** (*bool*) –Whether to use the output of encoder `out_enc` as *value* of attention layer. If False, the original feature `feat` will be used.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

警告: This decoder will not predict the final class which is assumed to be <*PAD*>. Therefore, its output size is always $C - 1$. <*PAD*> is also ignored by loss as specified in *mmocr.models.textrecog.recognizer.EncodeDecodeRecognizer*.

forward_test (*feat*, *out_enc*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A raw logit tensor of shape $(N, T, C-1)$ if `return_feature=False`. Otherwise it would be the hidden feature before the prediction projection layer, whose shape is (N, T, D_m).

返回类型 Tensor

forward_train (*feat*, *out_enc*, *targets_dict*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).

- **targets_dict** (*dict*) –A dict with the key `padded_targets`, a tensor of shape (N, T). Each element is the index of a character.
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A raw logit tensor of shape $(N, T, C-1)$ if `return_feature=False`. Otherwise it will be the hidden feature before the prediction projection layer, whose shape is (N, T, D_m).

返回类型 Tensor

class mmocr.models.textrecog.decoders.**RobustScannerDecoder**(*num_classes=None*, *dim_input=512*, *dim_model=128*, *max_seq_len=40*, *start_idx=0*, *mask=True*, *padding_idx=None*, *encode_value=False*, *hybrid_decoder=None*, *position_decoder=None*, *init_cfg=None*)

Decoder for RobustScanner.

RobustScanner: RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

参数

- **num_classes** (*int*) –Number of output classes C.
- **dim_input** (*int*) –Dimension D_i of input vector `feat`.
- **dim_model** (*int*) –Dimension D_m of the model. Should also be the same as encoder output vector `out_enc`.
- **max_seq_len** (*int*) –Maximum output sequence length T.
- **start_idx** (*int*) –The index of *<SOS>*.
- **mask** (*bool*) –Whether to mask input features according to `img_meta['valid_ratio']`.
- **padding_idx** (*int*) –The index of *<PAD>*.
- **encode_value** (*bool*) –Whether to use the output of encoder `out_enc` as *value* of attention layer. If False, the original feature `feat` will be used.
- **hybrid_decoder** (*dict*) –Configuration dict for hybrid decoder.
- **position_decoder** (*dict*) –Configuration dict for position decoder.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

警告: This decoder will not predict the final class which is assumed to be <*PAD*>. Therefore, its output size is always $C - 1$. <*PAD*> is also ignored by loss as specified in *mmocr.models.textrecog.recognizer.EncodeDecodeRecognizer*.

forward_test (*feat*, *out_enc*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 The output logit sequence tensor of shape $(N, T, C - 1)$.

返回类型 Tensor

forward_train (*feat*, *out_enc*, *targets_dict*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **targets_dict** (*dict*) –A dict with the key `padded_targets`, a tensor of shape (N, T). Each element is the index of a character.
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A raw logit tensor of shape $(N, T, C - 1)$.

返回类型 Tensor

class mmocr.models.textrecog.decoders.**SequenceAttentionDecoder**(*num_classes=None*, *rnn_layers=2*, *dim_input=512*, *dim_model=128*, *max_seq_len=40*, *start_idx=0*, *mask=True*, *padding_idx=None*, *dropout=0*, *return_feature=False*, *encode_value=False*, *init_cfg=None*)

Sequence attention decoder for RobustScanner.

RobustScanner: RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

参数

- **num_classes** (*int*) –Number of output classes C.
- **rnn_layers** (*int*) –Number of RNN layers.
- **dim_input** (*int*) –Dimension D_i of input vector `feat`.
- **dim_model** (*int*) –Dimension D_m of the model. Should also be the same as encoder output vector `out_enc`.
- **max_seq_len** (*int*) –Maximum output sequence length T.
- **start_idx** (*int*) –The index of *<SOS>*.
- **mask** (*bool*) –Whether to mask input features according to `img_meta['valid_ratio']`.
- **padding_idx** (*int*) –The index of *<PAD>*.
- **dropout** (*float*) –Dropout rate.
- **return_feature** (*bool*) –Return feature or logits as the result.
- **encode_value** (*bool*) –Whether to use the output of encoder `out_enc` as *value* of attention layer. If False, the original feature `feat` will be used.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

警告: This decoder will not predict the final class which is assumed to be *<PAD>*. Therefore, its output size is always $C - 1$. *<PAD>* is also ignored by loss as specified in *mmocr.models.textrecog.recognizer.EncodeDecodeRecognizer*.

forward_test(*feat*, *out_enc*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 The output logit sequence tensor of shape $(N, T, C-1)$.

返回类型 Tensor

forward_test_step(*feat*, *out_enc*, *decode_sequence*, *current_step*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **decode_sequence** (*Tensor*) –Shape (N, T). The tensor that stores history decoding result.
- **current_step** (*int*) –Current decoding step.
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 Shape $(N, C-1)$. The logit tensor of predicted tokens at current time step.

返回类型 Tensor

forward_train(*feat*, *out_enc*, *targets_dict*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **targets_dict** (*dict*) –A dict with the key `padded_targets`, a tensor of shape (N, T). Each element is the index of a character.
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A raw logit tensor of shape $(N, T, C-1)$ if `return_feature=False`. Otherwise it would be the hidden feature before the prediction projection layer, whose shape is (N, T, D_m).

返回类型 Tensor

class mmocr.models.textrecog.decoders.**SequentialSARDecoder**(*num_classes=37*, *enc_bi_rnn=False*, *dec_bi_rnn=False*, *dec_gru=False*, *d_k=64*, *d_model=512*, *d_enc=512*, *pred_dropout=0.0*, *mask=True*, *max_seq_len=40*, *start_idx=0*, *padding_idx=92*, *pred_concat=False*, *init_cfg=None*, ***kwargs*)

Implementation Sequential Decoder module in 'SAR.

<https://arxiv.org/abs/1811.00751>'_.

参数

- **num_classes** (*int*) –Output class number C.
- **enc_bi_rnn** (*bool*) –If True, use bidirectional RNN in encoder.
- **dec_bi_rnn** (*bool*) –If True, use bidirectional RNN in decoder.
- **dec_do_rnn** (*float*) –Dropout of RNN layer in decoder.
- **dec_gru** (*bool*) –If True, use GRU, else LSTM in decoder.
- **d_k** (*int*) –Dim of conv layers in attention module.
- **d_model** (*int*) –Dim of channels from backbone D_i.
- **d_enc** (*int*) –Dim of encoder RNN layer D_m.
- **pred_dropout** (*float*) –Dropout probability of prediction layer.
- **max_seq_len** (*int*) –Maximum sequence length during decoding.
- **mask** (*bool*) –If True, mask padding in feature map.
- **start_idx** (*int*) –Index of start token.
- **padding_idx** (*int*) –Index of padding token.
- **pred_concat** (*bool*) –If True, concat glimpse feature from attention with holistic feature and hidden state.

forward_test(*feat*, *out_enc*, *img_metas*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).

- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A raw logit tensor of shape $(N, T, C-1)$.

返回类型 Tensor

forward_train(*feat*, *out_enc*, *targets_dict*, *img_metas=None*)

参数

- **feat** (*Tensor*) –Tensor of shape (N, D_i, H, W).
- **out_enc** (*Tensor*) –Encoder output of shape (N, D_m, H, W).
- **targets_dict** (*dict*) –A dict with the key `padded_targets`, a tensor of shape (N, T). Each element is the index of a character.
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 A raw logit tensor of shape $(N, T, C-1)$.

返回类型 Tensor

24.14 textrecog_losses

class mmocr.models.textrecog.losses.**CELoss**(*ignore_index=-1*, *reduction='none'*)

Implementation of loss module for encoder-decoder based text recognition method with CrossEntropy loss.

参数

- **ignore_index** (*int*) –Specifies a target value that is ignored and does not contribute to the input gradient.
- **reduction** (*str*) –Specifies the reduction to apply to the output, should be one of the following: (“none” , “mean” , “sum”).

forward(*outputs*, *targets_dict*, *img_metas=None*)

参数

- **outputs** (*Tensor*) –A raw logit tensor of shape (N, T, C).
- **targets_dict** (*dict*) –A dict with a key `padded_targets`, which is a tensor of shape (N, T). Each element is the index of a character.
- **img_metas** (*None*) –Unused.

返回 A loss dict with the key `loss_ce`.

返回类型 dict

class mmocr.models.textrecog.losses.**CTCLoss**(*flatten=True*, *blank=0*, *reduction='mean'*, *zero_infinity=False*, ***kwargs*)

Implementation of loss module for CTC-loss based text recognition.

参数

- **flatten** (*bool*) –If True, use flattened targets, else padded targets.
- **blank** (*int*) –Blank label. Default 0.
- **reduction** (*str*) –Specifies the reduction to apply to the output, should be one of the following: ('none', 'mean', 'sum').
- **zero_infinity** (*bool*) –Whether to zero infinite losses and the associated gradients. Default: False. Infinite losses mainly occur when the inputs are too short to be aligned to the targets.

forward(*outputs*, *targets_dict*, *img_metas=None*)

参数

- **outputs** (*Tensor*) –A raw logit tensor of shape (N, T, C).
- **targets_dict** (*dict*) –A dict with 3 keys `target_lengths`, `flatten_targets` and `targets`.
  - `target_lengths` (Tensor): A tensor of shape (N). Each item is the length of a word.
  - `flatten_targets` (Tensor): Used if `self.flatten=True` (default). A tensor of shape (sum(targets_dict['target_lengths'])). Each item is the index of a character.
  - `targets` (Tensor): Used if `self.flatten=False`. A tensor of (N, T). Empty slots are padded with `self.blank`.
- **img_metas** (*dict*) –A dict that contains meta information of input images. Preferably with the key `valid_ratio`.

返回 The loss dict with key `loss_ctc`.

返回类型 dict

class mmocr.models.textrecog.losses.**SARLoss**(*ignore_index=0*, *reduction='mean'*, ***kwargs*)

Implementation of loss module in 'SAR.

<https://arxiv.org/abs/1811.00751>'_.

参数

- **ignore_index** (*int*) –Specifies a target value that is ignored and does not contribute to the input gradient.
- **reduction** (*str*) –Specifies the reduction to apply to the output, should be one of the following: (“none” , “mean” , “sum”).

警告: SARLoss assumes that the first input token is always *<SOS>*.

class mmocr.models.textrecog.losses.**SegLoss**(*seg_downsample_ratio=0.5*, *seg_with_loss_weight=True*, *ignore_index=255*, ***kwargs*)

Implementation of loss module for segmentation based text recognition method.

参数

- **seg_downsample_ratio** (*float*) –Downsample ratio of segmentation map.
- **seg_with_loss_weight** (*bool*) –If True, set weight for segmentation loss.
- **ignore_index** (*int*) –Specifies a target value that is ignored and does not contribute to the input gradient.

forward(*out_neck*, *out_head*, *gt_kernels*)

参数

- **out_neck** (*None*) –Unused.
- **out_head** (*Tensor*) –The output from head whose shape is (N, C, H, W).
- **gt_kernels** (*BitmapMasks*) –The ground truth masks.

返回 A loss dictionary with the key `loss_seg`.

返回类型 dict

class mmocr.models.textrecog.losses.**TFLoss**(*ignore_index=- 1*, *reduction='none'*, *flatten=True*, ***kwargs*)

Implementation of loss module for transformer.

参数

- **ignore_index** (*int, optional*) –The character index to be ignored in loss computation.
- **reduction** (*str*) –Type of reduction to apply to the output, should be one of the following: (“none” , “mean” , “sum”).
- **flatten** (*bool*) –Whether to flatten the vectors for loss computation.

警告： TFLoss assumes that the first input token is always <*SOS*>.

24.15 textrecog_backbones

class mmocr.models.textrecog.backbones.**NRTRModalityTransform**(*input_channels=3*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解： Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.backbones.**ResNet31OCR**(*base_channels=3*, *layers=[1, 2, 5, 3]*, *channels=[64, 128, 256, 256, 512, 512, 512]*, *out_indices=None*, *stage4_pool_cfg={'kernel_size': (2, 1), 'stride': (2, 1)}*, *last_stage_pool=False*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

Implement ResNet backbone for text recognition, modified from ResNet

参数

- **base_channels** (*int*) –Number of channels of input image tensor.
- **layers** (*list[int]*) –List of BasicBlock number for each stage.
- **channels** (*list[int]*) –List of out_channels of Conv2d layer.
- **out_indices** (*None | Sequence[int]*) –Indices of output stages.

- **stage4_pool_cfg** (*dict*) –Dictionary to construct and configure pooling layer in stage 4.
- **last_stage_pool** (*bool*) –If True, add *MaxPool2d* layer to last stage.

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.backbones.**ShallowCNN**(*input_channels=1*, *hidden_dim=512*, *init_cfg=[{'type': 'Kaiming', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

Implement Shallow CNN block for SATRN.

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

参数

- **base_channels** (*int*) –Number of channels of input image tensor D_i.
- **hidden_dim** (*int*) –Size of hidden layers of the model D_m.
- **init_cfg** (*dict or list[dict], optional*) –Initialization configs.

forward(*x*)

参数 **x** (*Tensor*) –Input image feature (N, D_i, H, W).

返回 A tensor of shape $(N, D_m, H/4, W/4)$.

返回类型 Tensor

class mmocr.models.textrecog.backbones.**VeryDeepVgg**(*leaky_relu=True*, *input_channels=3*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}, {'type': 'Uniform', 'layer': 'BatchNorm2d'}]*)

Implement VGG-VeryDeep backbone for text recognition, modified from VGG-VeryDeep

参数

- **leaky_relu** (*bool*) –Use leakyRelu or not.
- **input_channels** (*int*) –Number of channels of input image tensor.

forward(*x*)

参数 **x** (*Tensor*) –Images of shape (N, C, H, W).

返回 The feature Tensor of shape $(N, 512, H/32, (W/4+1))$.

返回类型 Tensor

24.16 textrecog_layers

class mmocr.models.textrecog.layers.**Adaptive2DPositionalEncoding**(*d_hid=512*, *n_height=100*, *n_width=100*, *dropout=0.1*, *init_cfg=[{'type': 'Xavier', 'layer': 'Conv2d'}]*)

Implement Adaptive 2D positional encoder for SATRN, see SATRN Modified from https://github.com/Media-Smart/vedastr Licensed under the Apache License, Version 2.0 (the "License");

参数

- **d_hid** (*int*) –Dimensions of hidden layer.
- **n_height** (*int*) –Max height of the 2D feature output.
- **n_width** (*int*) –Max width of the 2D feature output.
- **dropout** (*int*) –Size of hidden layers of the model.

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**BasicBlock**(*inplanes*, *planes*, *stride=1*, *downsample=False*)

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**BidirectionalLSTM**(*nIn*, *nHidden*, *nOut*)

forward(*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**Bottleneck**(*inplanes*, *planes*, *stride=1*, *downsample=False*)

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**DotProductAttentionLayer**(*dim_model=None*)

forward(*query*, *key*, *value*, *mask=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while

the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**PositionAwareLayer**(*dim_model*, *rnn_layers=2*)

forward(*img_feature*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mmocr.models.textrecog.layers.**RobustScannerFusionLayer**(*dim_model*, *dim=- 1*, *init_cfg=None*)

forward(*x0*, *x1*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

24.17 kie_extractors

class mmocr.models.kie.extractors.**SDMGR**(*backbone*, *neck=None*, *bbox_head=None*, *extractor={'featmap_strides': [1], 'roi_layer': {'output_size': 7, 'type': 'RoIAlign'}, 'type': 'mmdet.SingleRoIExtractor'}*, *visual_modality=False*, *train_cfg=None*, *test_cfg=None*, *class_list=None*, *init_cfg=None*, *openset=False*)

The implementation of the paper: Spatial Dual-Modality Graph Reasoning for Key Information Extraction. https://arxiv.org/abs/2103.14470.

参数

- **visual_modality** (*bool*) –Whether use the visual modality.

- **class_list** (*None | str*) –Mapping file of class index to class name. If None, class index will be shown in *show_results*, else class name.

extract_feat(*img*, *gt_bboxes*)

Directly extract features from the backbone+neck.

forward_test(*img*, *img_metas*, *relations*, *texts*, *gt_bboxes*, *rescale=False*)

Args: imgs (List[Tensor]): the outer list indicates test-time

augmentations and inner Tensor should have a shape NxCxHxW, which contains all images in the batch.

img_metas (List[List[dict]]): the outer list indicates test-time augs (multiscale, flip, etc.) and the inner list indicates images in a batch.

forward_train(*img*, *img_metas*, *relations*, *texts*, *gt_bboxes*, *gt_labels*)

参数

- **img** (*tensor*) –Input images of shape (N, C, H, W). Typically these should be mean centered and std scaled.
- **img_metas** (*list[dict]*) –A list of image info dict where each dict contains: 'img_shape', 'scale_factor', 'flip', and may also contain 'filename', 'ori_shape', 'pad_shape', and 'img_norm_cfg'. For details of the values of these keys, please see `mmdet.datasets.pipelines.Collect`.
- **relations** (*list[tensor]*) –Relations between bboxes.
- **texts** (*list[tensor]*) –Texts in bboxes.
- **gt_bboxes** (*list[tensor]*) –Each item is the truth boxes for each image in [tl_x, tl_y, br_x, br_y] format.
- **gt_labels** (*list[tensor]*) –Class indices corresponding to each box.

返回 A dictionary of loss components.

返回类型 dict[str, tensor]

show_result(*img*, *result*, *boxes*, *win_name=''*, *show=False*, *wait_time=0*, *out_file=None*, ***kwargs*)

Draw *result* on *img*.

参数

- **img** (*str or tensor*) –The image to be displayed.
- **result** (*dict*) –The results to draw on *img*.
- **boxes** (*list*) –Bbox of img.
- **win_name** (*str*) –The window name.

- **wait_time** (*int*) –Value of waitKey param. Default: 0.
- **show** (*bool*) –Whether to show the image. Default: False.
- **out_file** (*str or None*) –The output filename. Default: None.

返回 Only if not *show* or *out_file*.

返回类型 img (tensor)

24.18 kie_heads

class mmocr.models.kie.heads.**SDMGRHead**(*num_chars=92*, *visual_dim=64*, *fusion_dim=1024*, *node_input=32*, *node_embed=256*, *edge_input=5*, *edge_embed=256*, *num_gnn=2*, *num_classes=26*, *loss={'type': 'SDMGRLoss'}*, *bidirectional=False*, *train_cfg=None*, *test_cfg=None*, *init_cfg={'mean': 0, 'override': {'name': 'edge_embed'}, 'std': 0.01, 'type': 'Normal'}*)

forward(*relations*, *texts*, *x=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

24.19 kie_losses

class mmocr.models.kie.losses.**SDMGRLoss**(*node_weight=1.0*, *edge_weight=1.0*, *ignore=- 100*)

The implementation the loss of key information extraction proposed in the paper: Spatial Dual-Modality Graph Reasoning for Key Information Extraction.

https://arxiv.org/abs/2103.14470.

forward(*node_preds*, *edge_preds*, *gts*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

注解： Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

CHAPTER 25

mmocr.datasets

class mmocr.datasets.**BaseDataset**(*ann_file*, *loader*, *pipeline*, *img_prefix=''*, *test_mode=False*)

Custom dataset for text detection, text recognition, and their downstream tasks.

1. The text detection annotation format is as follows: The *annotations* field is optional for testing (this is one line of anno_file, with line-json-str

   converted to dict for visualizing only).

   {
   "file_name": "sample.jpg",
   "height": 1080,
   "width": 960,
   "annotations":
   [
   {
   "iscrowd": 0,
   "category_id": 1,
   "bbox": [357.0, 667.0, 804.0, 100.0],
   "segmentation": [[361, 667, 710, 670,
   72, 767, 357, 763]]
   }
   ]
   }

2. The two text recognition annotation formats are as follows: The *x1,y1,x2,y2,x3,y3,x4,y4* field is used for online crop augmentation during training.

   format1: sample.jpg hello format2: sample.jpg 20 20 100 20 100 40 20 40 hello

参数

- **ann_file** (*str*) –Annotation file path.

- **pipeline** (*list[dict]*) –Processing pipeline.
- **loader** (*dict*) –Dictionary to construct loader to load annotation infos.
- **img_prefix** (*str, optional*) –Image prefix to generate full image path.
- **test_mode** (*bool, optional*) –If set True, try…except will be turned off in __getitem__.

evaluate(*results*, *metric=None*, *logger=None*, ***kwargs*)

Evaluate the dataset.

参数

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

返回 float]

返回类型 dict[str

format_results(*results*, ***kwargs*)

Placeholder to format result to dataset-specific output.

pre_pipeline(*results*)

Prepare results dict for pipeline.

prepare_test_img(*img_info*)

Get testing data from pipeline.

参数 **idx** (*int*) –Index of data.

返回

Testing data after pipeline with new keys introduced by pipeline.

返回类型 dict

prepare_train_img(*index*)

Get training data and annotations from pipeline.

参数 **index** (*int*) –Index of data.

返回

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

返回类型 dict

class mmocr.datasets.**CustomFormatBundle**(*keys=[]*, *call_super=True*, *visualize={'boundary_key': None, 'flag': False}*)

Custom formatting bundle.

It formats common fields such as ‘img’ and ‘proposals’ as done in DefaultFormatBundle, while other fields such as ‘gt_kernels’ and ‘gt_effective_region_mask’ will be formatted to DC as follows:

- gt_kernels: to DataContainer (cpu_only=True)
- gt_effective_mask: to DataContainer (cpu_only=True)

参数

- **keys** (*list[str]*) –Fields to be formatted to DC only.
- **call_super** (*bool*) –If True, format common fields by DefaultFormatBundle, else format fields in keys above only.
- **visualize** (*dict*) –If flag=True, visualize gt mask for debugging.

class mmocr.datasets.**DBNetTargets**(*shrink_ratio=0.4*, *thr_min=0.3*, *thr_max=0.7*, *min_short_size=8*)

Generate gt shrunk text, gt threshold map, and their effective region masks to learn DBNet: Real-time Scene Text Detection with Differentiable Binarization [https://arxiv.org/abs/1911.08947]. This was partially adapted from https://github.com/MhLiao/DB.

参数

- **shrink_ratio** (*float*) –The area shrunk ratio between text kernels and their text masks.
- **thr_min** (*float*) –The minimum value of the threshold map.
- **thr_max** (*float*) –The maximum value of the threshold map.
- **min_short_size** (*int*) –The minimum size of polygon below which the polygon is invalid.

draw_border_map(*polygon*, *canvas*, *mask*)

Generate threshold map for one polygon.

参数

- **polygon** (*ndarray*) –The polygon boundary ndarray.
- **canvas** (*ndarray*) –The generated threshold map.
- **mask** (*ndarray*) –The generated threshold mask.

find_invalid(*results*)

Find invalid polygons.

参数 **results** (*dict*) –The dict containing gt_mask.

返回 The indicators for ignoring polygons.

返回类型 ignore_tags (list[bool])

generate_targets(*results*)

Generate the gt targets for DBNet.

参数 **results** (*dict*) –The input result dictionary.

返回 The output result dictionary.

返回类型 results (dict)

generate_thr_map(*img_size*, *polygons*)

Generate threshold map.

参数

- **img_size** (*tuple(int)*) –The image size (h,w)
- **polygons** (*list(ndarray)*) –The polygon list.

返回 The generated threshold map. thr_mask (ndarray): The effective mask of threshold map.

返回类型 thr_map (ndarray)

ignore_texts(*results*, *ignore_tags*)

Ignore gt masks and gt_labels while padding gt_masks_ignore in results given ignore_tags.

参数

- **results** (*dict*) –Result for one image.
- **ignore_tags** (*list[int]*) –Indicate whether to ignore its corresponding ground truth text.

返回 Results after filtering.

返回类型 results (dict)

invalid_polygon(*poly*)

Judge the input polygon is invalid or not. It is invalid if its area smaller than 1 or the shorter side of its minimum bounding box smaller than min_short_size.

参数 **poly** (*ndarray*) –The polygon boundary point sequence.

返回 Whether the polygon is invalid.

返回类型 True/False (bool)

class mmocr.datasets.**FCENetTargets**(*fourier_degree=5*, *resample_step=4.0*, *center_region_shrink_ratio=0.3*, *level_size_divisors=(8, 16, 32)*, *level_proportion_range=((0, 0.4), (0.3, 0.7), (0.6, 1.0))*)

Generate the ground truth targets of FCENet: Fourier Contour Embedding for Arbitrary-Shaped Text Detection.

[https://arxiv.org/abs/2104.10442]

参数

- **fourier_degree** (*int*) –The maximum Fourier transform degree k.
- **resample_step** (*float*) –The step size for resampling the text center line (TCL). It's better not to exceed half of the minimum width.
- **center_region_shrink_ratio** (*float*) –The shrink ratio of text center region.

- **level_size_divisors** (*tuple(int)*) –The downsample ratio on each level.
- **level_proportion_range** (*tuple(tuple(int))*) –The range of text sizes assigned to each level.

cal_fourier_signature (*polygon*, *fourier_degree*)

Calculate Fourier signature from input polygon.

参数

- **polygon** (*ndarray*) –The input polygon.
- **fourier_degree** (*int*) –The maximum Fourier degree K.

返回

An array shaped (2k+1, 2) containing real part and image part of 2k+1 Fourier coefficients.

返回类型 fourier_signature (ndarray)

clockwise (*c*, *fourier_degree*)

Make sure the polygon reconstructed from Fourier coefficients c in the clockwise direction.

参数 **polygon** (*list[float]*) –The origin polygon.

返回 The polygon in clockwise point order.

返回类型 new_polygon (lost[float])

generate_center_region_mask (*img_size*, *text_polys*)

Generate text center region mask.

参数

- **img_size** (*tuple*) –The image size of (height, width).
- **text_polys** (*list[list[ndarray]]*) –The list of text polygons.

返回 The text center region mask.

返回类型 center_region_mask (ndarray)

generate_fourier_maps (*img_size*, *text_polys*)

Generate Fourier coefficient maps.

参数

- **img_size** (*tuple*) –The image size of (height, width).
- **text_polys** (*list[list[ndarray]]*) –The list of text polygons.

返回

The Fourier coefficient real part maps. fourier_image_map (ndarray): The Fourier coefficient image part

maps.

返回类型 fourier_real_map (ndarray)

generate_level_targets(*img_size*, *text_polys*, *ignore_polys*)

Generate ground truth target on each level.

参数

- **img_size** (*list[int]*) –Shape of input image.
- **text_polys** (*list[list[ndarray]]*) –A list of ground truth polygons.
- **ignore_polys** (*list[list[ndarray]]*) –A list of ignored polygons.

返回 A list of ground target on each level.

返回类型 level_maps (list(ndarray))

generate_targets(*results*)

Generate the ground truth targets for FCENet.

参数 **results** (*dict*) –The input result dictionary.

返回 The output result dictionary.

返回类型 results (dict)

normalize_polygon(*polygon*)

Normalize one polygon so that its start point is at right most.

参数 **polygon** (*list[float]*) –The origin polygon.

返回 The polygon with start point at right.

返回类型 new_polygon (lost[float])

poly2fourier(*polygon*, *fourier_degree*)

Perform Fourier transformation to generate Fourier coefficients ck from polygon.

参数

- **polygon** (*ndarray*) –An input polygon.
- **fourier_degree** (*int*) –The maximum Fourier degree K.

返回 Fourier coefficients.

返回类型 c (ndarray(complex))

resample_polygon(*polygon*, *n=400*)

Resample one polygon with n points on its boundary.

参数

- **polygon** (*list[float]*) –The input polygon.
- **n** (*int*) –The number of resampled points.

返回 The resampled polygon.

返回类型 resampled_polygon (list[float])

class mmocr.datasets.**HardDiskLoader**(*ann_file*, *parser*, *repeat=1*)

Load annotation file from hard disk to RAM.

参数 **ann_file** (*str*) –Annotation file path.

class mmocr.datasets.**IcdarDataset**(*ann_file*, *pipeline*, *classes=None*, *data_root=None*, *img_prefix=''*, *seg_prefix=None*, *proposal_file=None*, *test_mode=False*, *filter_empty_gt=True*, *select_first_k=-1*)

evaluate(*results*, *metric='hmean-iou'*, *logger=None*, *score_thr=0.3*, *rank_list=None*, ***kwargs*)

Evaluate the hmean metric.

参数

- **results** (*list[dict]*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.
- **rank_list** (*str*) –json file used to save eval result of each image after ranking.

返回 float]]: The evaluation results.

返回类型 dict[dict[str

load_annotations(*ann_file*)

Load annotation from COCO style annotation file.

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

返回 Annotation info from COCO api.

返回类型 list[dict]

class mmocr.datasets.**KIEDataset**(*ann_file=None*, *loader=None*, *dict_file=None*, *img_prefix=''*, *pipeline=None*, *norm=10.0*, *directed=False*, *test_mode=True*, ***kwargs*)

参数

- **ann_file** (*str*) –Annotation file path.
- **pipeline** (*list[dict]*) –Processing pipeline.
- **loader** (*dict*) –Dictionary to construct loader to load annotation infos.
- **img_prefix** (*str, optional*) –Image prefix to generate full image path.

- **test_mode** (*bool, optional*) –If True, try…except will be turned off in __getitem__.
- **dict_file** (*str*) –Character dict file path.
- **norm** (*float*) –Norm to map value from one range to another.

compute_relation(*boxes*)

Compute relation between every two boxes.

evaluate(*results*, *metric='macro_f1'*, *metric_options={'macro_f1': {'ignores': []}}*, ***kwargs*)

Evaluate the dataset.

参数

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

返回 float]

返回类型 dict[str

list_to_numpy(*ann_infos*)

Convert bboxes, relations, texts and labels to ndarray.

pad_text_indices(*text_inds*)

Pad text index to same length.

pre_pipeline(*results*)

Prepare results dict for pipeline.

prepare_train_img(*index*)

Get training data and annotations from pipeline.

参数 **index** (*int*) –Index of data.

返回

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

返回类型 dict

class mmocr.datasets.**LineJsonParser**(*keys=[]*)

Parse json-string of one line in annotation file to dict format.

参数 **keys** (*list[str]*) –Keys in both json-string and result dict.

class mmocr.datasets.**LineStrParser**(*keys=['filename', 'text']*, *keys_idx=[0, 1]*, *separator=' '*, ***kwargs*)

Parse string of one line in annotation file to dict format.

参数

- **keys** (*list[str]*) –Keys in result dict.

- **keys_idx** (*list[int]*) –Value index in sub-string list for each key above.
- **separator** (*str*) –Separator to separate string to list of sub-string.

class mmocr.datasets.**LmdbLoader**(*ann_file*, *parser*, *repeat=1*)

Load annotation file with lmdb storage backend.

class mmocr.datasets.**NerDataset**(*ann_file*, *loader*, *pipeline*, *img_prefix=''*, *test_mode=False*)

Custom dataset for named entity recognition tasks.

参数

- **ann_file** (*txt*) –Annotation file path.
- **loader** (*dict*) –Dictionary to construct loader to load annotation infos.
- **pipeline** (*list[dict]*) –Processing pipeline.
- **test_mode** (*bool, optional*) –If True, try…except will be turned off in __getitem__.

evaluate(*results*, *metric=None*, *logger=None*, ***kwargs*)

Evaluate the dataset.

参数

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

返回

A dict containing the following keys: ’acc’, ‘recall’, ‘f1-score’.

返回类型 info (dict)

prepare_train_img(*index*)

Get training data and annotations after pipeline.

参数 **index** (*int*) –Index of data.

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

返回类型 dict

class mmocr.datasets.**OCRDataset**(*ann_file*, *loader*, *pipeline*, *img_prefix=''*, *test_mode=False*)

evaluate(*results*, *metric='acc'*, *logger=None*, ***kwargs*)

Evaluate the dataset.

参数

- **results** (*list*) –Testing results of the dataset.

- **metric** (*str | list[str]*) –Metrics to be evaluated.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.

返回 float]

返回类型 dict[str

pre_pipeline(*results*)

Prepare results dict for pipeline.

class mmocr.datasets.**OCRSegDataset**(*ann_file*, *loader*, *pipeline*, *img_prefix=''*, *test_mode=False*)

pre_pipeline(*results*)

Prepare results dict for pipeline.

prepare_train_img(*index*)

Get training data and annotations from pipeline.

参数 **index** (*int*) –Index of data.

返回

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

返回类型 dict

class mmocr.datasets.**OpensetKIEDataset**(*ann_file*, *loader*, *dict_file*, *img_prefix=''*, *pipeline=None*, *norm=10.0*, *link_type='one-to-one'*, *edge_thr=0.5*, *test_mode=True*, *key_node_idx=1*, *value_node_idx=2*, *node_classes=4*)

Openset KIE classifies the nodes (i.e. text boxes) into bg/key/value categories, and additionally learns key-value relationship among nodes.

参数

- **ann_file** (*str*) –Annotation file path.
- **loader** (*dict*) –Dictionary to construct loader to load annotation infos.
- **dict_file** (*str*) –Character dict file path.
- **img_prefix** (*str, optional*) –Image prefix to generate full image path.
- **pipeline** (*list[dict]*) –Processing pipeline.
- **norm** (*float*) –Norm to map value from one range to another.
- **link_type** (*str*) –`one-to-one` | `one-to-many` | `many-to-one` | `many-to-many`. For `many-to-many`, one key box can have many values and vice versa.
- **edge_thr** (*float*) –Score threshold for a valid edge.

- **test_mode** (*bool, optional*) –If True, try…except will be turned off in __getitem__.
- **key_node_idx** (*int*) –Index of key in node classes.
- **value_node_idx** (*int*) –Index of value in node classes.
- **node_classes** (*int*) –Number of node classes.

compute_openset_f1(*preds*, *gts*)

Compute openset macro-f1 and micro-f1 score.

参数

- **preds** –(list[dict]): List of prediction results, including keys: `filename`, `pairs`, etc.
- **gts** –(list[dict]): List of ground-truth infos, including keys: `filename`, `pairs`, etc.

返回 Evaluation result with keys: `node_openset_micro_f1`, `node_openset_macro_f1`, `edge_openset_f1`.

返回类型 dict

decode_gt(*filename*)

Decode ground truth.

Assemble boxes and labels into bboxes.

decode_pred(*result*)

Decode prediction.

Assemble boxes and predicted labels into bboxes, and convert edges into matrix.

evaluate(*results*, *metric='openset_f1'*, *metric_options=None*, ***kwargs*)

Evaluate the dataset.

参数

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

返回 float]

返回类型 dict[str

list_to_numpy(*ann_infos*)

Convert bboxes, relations, texts and labels to ndarray.

pre_pipeline(*results*)

Prepare results dict for pipeline.

class mmocr.datasets.**TextDetDataset**(*ann_file*, *loader*, *pipeline*, *img_prefix=''*, *test_mode=False*)

evaluate(*results*, *metric='hmean-iou'*, *score_thr=0.3*, *rank_list=None*, *logger=None*, ***kwargs*)

Evaluate the dataset.

参数

- **results** (*list*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated.
- **score_thr** (*float*) –Score threshold for prediction map.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.
- **rank_list** (*str*) –json file used to save eval result of each image after ranking.

返回 float]

返回类型 dict[str

prepare_train_img(*index*)

Get training data and annotations from pipeline.

参数 **index** (*int*) –Index of data.

返回

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

返回类型 dict

class mmocr.datasets.**UniformConcatDataset**(*datasets*, *separate_eval=True*, *pipeline=None*, ***kwargs*)

A wrapper of concatenated dataset.

Same as `torch.utils.data.dataset.ConcatDataset`, but concat the group flag for image aspect ratio.

参数

- **datasets** (list[`Dataset`]) –A list of datasets.
- **separate_eval** (*bool*) –Whether to evaluate the results separately if it is used as validation dataset. Defaults to True.

mmocr.datasets.**build_dataloader**(*dataset*, *samples_per_gpu*, *workers_per_gpu*, *num_gpus=1*, *dist=True*, *shuffle=True*, *seed=None*, *runner_type='EpochBasedRunner'*, *persistent_workers=False*, ***kwargs*)

Build PyTorch DataLoader.

In distributed training, each GPU/process has a dataloader. In non-distributed training, there is only one dataloader for all GPUs.

参数

- **dataset** (*Dataset*) –A PyTorch dataset.
- **samples_per_gpu** (*int*) –Number of training samples on each GPU, i.e., batch size of each GPU.
- **workers_per_gpu** (*int*) –How many subprocesses to use for data loading for each GPU.
- **num_gpus** (*int*) –Number of GPUs. Only used in non-distributed training.
- **dist** (*bool*) –Distributed training/test or not. Default: True.
- **shuffle** (*bool*) –Whether to shuffle the data at every epoch. Default: True.
- **seed** (*int, Optional*) –Seed to be used. Default: None.
- **runner_type** (*str*) –Type of runner. Default: *EpochBasedRunner*
- **persistent_workers** (*bool*) –If True, the data loader will not shutdown the worker processes after a dataset has been consumed once. This allows to maintain the workers *Dataset* instances alive. This argument is only valid when PyTorch>=1.7.0. Default: False.
- **kwargs** –any keyword argument to be used to initialize DataLoader

返回 A PyTorch dataloader.

返回类型 DataLoader

25.1 datasets

class mmocr.datasets.base_dataset.**BaseDataset**(*ann_file*, *loader*, *pipeline*, *img_prefix=''*, *test_mode=False*)

Custom dataset for text detection, text recognition, and their downstream tasks.

1. The text detection annotation format is as follows: The *annotations* field is optional for testing (this is one line of anno_file, with line-json-str

   converted to dict for visualizing only).

   {
   "file_name": "sample.jpg",
   "height": 1080,
   "width": 960,
   "annotations":

   [

   {
   "iscrowd": 0,
   "category_id": 1,
   "bbox": [357.0, 667.0, 804.0, 100.0],
   "segmentation": [[361, 667, 710, 670,

   72, 767, 357, 763]]

```
                }
            ]
        }
```

2. The two text recognition annotation formats are as follows: The *x1,y1,x2,y2,x3,y3,x4,y4* field is used for online crop augmentation during training.

   format1: sample.jpg hello format2: sample.jpg 20 20 100 20 100 40 20 40 hello

   参数

   - **ann_file** (*str*) –Annotation file path.
   - **pipeline** (*list[dict]*) –Processing pipeline.
   - **loader** (*dict*) –Dictionary to construct loader to load annotation infos.
   - **img_prefix** (*str, optional*) –Image prefix to generate full image path.
   - **test_mode** (*bool, optional*) –If set True, try…except will be turned off in __getitem__.

evaluate(*results*, *metric=None*, *logger=None*, ***kwargs*)

Evaluate the dataset.

参数

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

返回 float]

返回类型 dict[str

format_results(*results*, ***kwargs*)

Placeholder to format result to dataset-specific output.

pre_pipeline(*results*)

Prepare results dict for pipeline.

prepare_test_img(*img_info*)

Get testing data from pipeline.

参数 **idx** (*int*) –Index of data.

返回

Testing data after pipeline with new keys introduced by pipeline.

返回类型 dict

prepare_train_img(*index*)

Get training data and annotations from pipeline.

参数 **index** (*int*) –Index of data.

返回

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

返回类型 dict

class mmocr.datasets.icdar_dataset.**IcdarDataset**(*ann_file*, *pipeline*, *classes=None*, *data_root=None*, *img_prefix=''*, *seg_prefix=None*, *proposal_file=None*, *test_mode=False*, *filter_empty_gt=True*, *select_first_k=-1*)

evaluate(*results*, *metric='hmean-iou'*, *logger=None*, *score_thr=0.3*, *rank_list=None*, ***kwargs*)

Evaluate the hmean metric.

参数

- **results** (*list[dict]*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.
- **rank_list** (*str*) –json file used to save eval result of each image after ranking.

返回 float]]: The evaluation results.

返回类型 dict[dict[str

load_annotations(*ann_file*)

Load annotation from COCO style annotation file.

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

返回 Annotation info from COCO api.

返回类型 list[dict]

class mmocr.datasets.ocr_dataset.**OCRDataset**(*ann_file*, *loader*, *pipeline*, *img_prefix=''*, *test_mode=False*)

evaluate(*results*, *metric='acc'*, *logger=None*, ***kwargs*)

Evaluate the dataset.

参数

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

返回 float]

返回类型 dict[str

pre_pipeline(*results*)

Prepare results dict for pipeline.

class mmocr.datasets.ocr_seg_dataset.**OCRSegDataset**(*ann_file*, *loader*, *pipeline*, *img_prefix=''*, *test_mode=False*)

pre_pipeline(*results*)

Prepare results dict for pipeline.

prepare_train_img(*index*)

Get training data and annotations from pipeline.

参数 **index** (*int*) –Index of data.

返回

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

返回类型 dict

class mmocr.datasets.text_det_dataset.**TextDetDataset**(*ann_file*, *loader*, *pipeline*, *img_prefix=''*, *test_mode=False*)

evaluate(*results*, *metric='hmean-iou'*, *score_thr=0.3*, *rank_list=None*, *logger=None*, ***kwargs*)

Evaluate the dataset.

参数

- **results** (*list*) –Testing results of the dataset.
- **metric** (*str | list[str]*) –Metrics to be evaluated.
- **score_thr** (*float*) –Score threshold for prediction map.
- **logger** (*logging.Logger | str | None*) –Logger used for printing related information during evaluation. Default: None.
- **rank_list** (*str*) –json file used to save eval result of each image after ranking.

返回 float]

返回类型 dict[str

prepare_train_img(*index*)

Get training data and annotations from pipeline.

参数 **index** (*int*) –Index of data.

返回

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

返回类型 dict

class mmocr.datasets.kie_dataset.**KIEDataset**(*ann_file=None*, *loader=None*, *dict_file=None*, *img_prefix=''*, *pipeline=None*, *norm=10.0*, *directed=False*, *test_mode=True*, ***kwargs*)

参数

- **ann_file** (*str*) –Annotation file path.
- **pipeline** (*list[dict]*) –Processing pipeline.
- **loader** (*dict*) –Dictionary to construct loader to load annotation infos.
- **img_prefix** (*str, optional*) –Image prefix to generate full image path.
- **test_mode** (*bool, optional*) –If True, try…except will be turned off in __getitem__.
- **dict_file** (*str*) –Character dict file path.
- **norm** (*float*) –Norm to map value from one range to another.

compute_relation(*boxes*)

Compute relation between every two boxes.

evaluate(*results*, *metric='macro_f1'*, *metric_options={'macro_f1': {'ignores': []}}*, ***kwargs*)

Evaluate the dataset.

参数

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

返回 float]

返回类型 dict[str

list_to_numpy(*ann_infos*)

Convert bboxes, relations, texts and labels to ndarray.

pad_text_indices(*text_inds*)

Pad text index to same length.

pre_pipeline(*results*)

Prepare results dict for pipeline.

prepare_train_img(*index*)

Get training data and annotations from pipeline.

参数 **index** (*int*) –Index of data.

返回

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

返回类型 dict

25.2 pipelines

class mmocr.datasets.pipelines.**ColorJitter**(***kwargs*)

An interface for torch color jitter so that it can be invoked in mmdetection pipeline.

class mmocr.datasets.pipelines.**CustomFormatBundle**(*keys=[]*, *call_super=True*, *visualize={'boundary_key': None, 'flag': False}*)

Custom formatting bundle.

It formats common fields such as ‘img’ and ‘proposals’ as done in DefaultFormatBundle, while other fields such as ‘gt_kernels’ and ‘gt_effective_region_mask’ will be formatted to DC as follows:

- gt_kernels: to DataContainer (cpu_only=True)
- gt_effective_mask: to DataContainer (cpu_only=True)

参数

- **keys** (*list[str]*) –Fields to be formatted to DC only.
- **call_super** (*bool*) –If True, format common fields by DefaultFormatBundle, else format fields in keys above only.
- **visualize** (*dict*) –If flag=True, visualize gt mask for debugging.

class mmocr.datasets.pipelines.**DBNetTargets**(*shrink_ratio=0.4*, *thr_min=0.3*, *thr_max=0.7*, *min_short_size=8*)

Generate gt shrunk text, gt threshold map, and their effective region masks to learn DBNet: Real-time Scene Text Detection with Differentiable Binarization [https://arxiv.org/abs/1911.08947]. This was partially adapted from https://github.com/MhLiao/DB.

参数

- **shrink_ratio** (*float*) –The area shrunk ratio between text kernels and their text masks.
- **thr_min** (*float*) –The minimum value of the threshold map.
- **thr_max** (*float*) –The maximum value of the threshold map.
- **min_short_size** (*int*) –The minimum size of polygon below which the polygon is invalid.

draw_border_map(*polygon*, *canvas*, *mask*)

Generate threshold map for one polygon.

参数

- **polygon** (*ndarray*) –The polygon boundary ndarray.
- **canvas** (*ndarray*) –The generated threshold map.
- **mask** (*ndarray*) –The generated threshold mask.

find_invalid(*results*)

Find invalid polygons.

参数 **results** (*dict*) –The dict containing gt_mask.

返回 The indicators for ignoring polygons.

返回类型 ignore_tags (list[bool])

generate_targets(*results*)

Generate the gt targets for DBNet.

参数 **results** (*dict*) –The input result dictionary.

返回 The output result dictionary.

返回类型 results (dict)

generate_thr_map(*img_size*, *polygons*)

Generate threshold map.

参数

- **img_size** (*tuple(int)*) –The image size (h,w)
- **polygons** (*list(ndarray)*) –The polygon list.

返回 The generated threshold map. thr_mask (ndarray): The effective mask of threshold map.

返回类型 thr_map (ndarray)

ignore_texts(*results*, *ignore_tags*)

Ignore gt masks and gt_labels while padding gt_masks_ignore in results given ignore_tags.

参数

- **results** (*dict*) –Result for one image.
- **ignore_tags** (*list[int]*) –Indicate whether to ignore its corresponding ground truth text.

返回 Results after filtering.

返回类型 results (dict)

invalid_polygon(*poly*)

Judge the input polygon is invalid or not. It is invalid if its area smaller than 1 or the shorter side of its minimum bounding box smaller than min_short_size.

参数 **poly** (*ndarray*) –The polygon boundary point sequence.

返回 Whether the polygon is invalid.

返回类型 True/False (bool)

class mmocr.datasets.pipelines.**FCENetTargets**(*fourier_degree=5*, *resample_step=4.0*, *center_region_shrink_ratio=0.3*, *level_size_divisors=(8, 16, 32)*, *level_proportion_range=((0, 0.4), (0.3, 0.7), (0.6, 1.0))*)

Generate the ground truth targets of FCENet: Fourier Contour Embedding for Arbitrary-Shaped Text Detection.

[https://arxiv.org/abs/2104.10442]

参数

- **fourier_degree** (*int*) –The maximum Fourier transform degree k.
- **resample_step** (*float*) –The step size for resampling the text center line (TCL). It's better not to exceed half of the minimum width.
- **center_region_shrink_ratio** (*float*) –The shrink ratio of text center region.
- **level_size_divisors** (*tuple(int)*) –The downsample ratio on each level.
- **level_proportion_range** (*tuple(tuple(int))*) –The range of text sizes assigned to each level.

cal_fourier_signature(*polygon*, *fourier_degree*)

Calculate Fourier signature from input polygon.

参数

- **polygon** (*ndarray*) –The input polygon.
- **fourier_degree** (*int*) –The maximum Fourier degree K.

返回

An array shaped (2k+1, 2) containing real part and image part of 2k+1 Fourier coefficients.

返回类型 fourier_signature (ndarray)

clockwise(*c*, *fourier_degree*)

Make sure the polygon reconstructed from Fourier coefficients c in the clockwise direction.

参数 **polygon** (*list[float]*) –The origin polygon.

返回 The polygon in clockwise point order.

返回类型 new_polygon (lost[float])

generate_center_region_mask(*img_size*, *text_polys*)

Generate text center region mask.

参数

- **img_size** (*tuple*) –The image size of (height, width).
- **text_polys** (*list[list[ndarray]]*) –The list of text polygons.

返回 The text center region mask.

返回类型 center_region_mask (ndarray)

generate_fourier_maps(*img_size*, *text_polys*)

Generate Fourier coefficient maps.

参数

- **img_size** (*tuple*) –The image size of (height, width).
- **text_polys** (*list[list[ndarray]]*) –The list of text polygons.

返回

The Fourier coefficient real part maps. fourier_image_map (ndarray): The Fourier coefficient image part

maps.

返回类型 fourier_real_map (ndarray)

generate_level_targets(*img_size*, *text_polys*, *ignore_polys*)

Generate ground truth target on each level.

参数

- **img_size** (*list[int]*) –Shape of input image.
- **text_polys** (*list[list[ndarray]]*) –A list of ground truth polygons.
- **ignore_polys** (*list[list[ndarray]]*) –A list of ignored polygons.

返回 A list of ground target on each level.

返回类型 level_maps (list(ndarray))

generate_targets(*results*)

Generate the ground truth targets for FCENet.

参数 **results** (*dict*) –The input result dictionary.

返回 The output result dictionary.

返回类型 results (dict)

normalize_polygon(*polygon*)

Normalize one polygon so that its start point is at right most.

参数 **polygon** (*list[float]*) –The origin polygon.

返回 The polygon with start point at right.

返回类型 new_polygon (lost[float])

poly2fourier(*polygon*, *fourier_degree*)

Perform Fourier transformation to generate Fourier coefficients ck from polygon.

参数

- **polygon** (*ndarray*) –An input polygon.
- **fourier_degree** (*int*) –The maximum Fourier degree K.

返回 Fourier coefficients.

返回类型 c (ndarray(complex))

resample_polygon(*polygon*, *n=400*)

Resample one polygon with n points on its boundary.

参数

- **polygon** (*list[float]*) –The input polygon.
- **n** (*int*) –The number of resampled points.

返回 The resampled polygon.

返回类型 resampled_polygon (list[float])

class mmocr.datasets.pipelines.**FancyPCA**(*eig_vec=None*, *eig_val=None*)

Implementation of PCA based image augmentation, proposed in the paper `Imagenet Classification With Deep Convolutional Neural Networks`.

It alters the intensities of RGB values along the principal components of ImageNet dataset.

class mmocr.datasets.pipelines.**ImgAug**(*args=None*)

A wrapper to use imgaug https://github.com/aleju/imgaug.

参数 **args** (*[list[list|dict]]*) –The argumentation list. For details, please refer to imgaug document. Take args=[[‘Fliplr’, 0.5], dict(cls=’Affine’, rotate=[-10, 10]), [‘Resize’, [0.5, 3.0]]] as an example. The args horizontally flip images with probability 0.5, followed by

random rotation with angles in range [-10, 10], and resize with an independent scale in range [0.5, 3.0] for each side of images.

class mmocr.datasets.pipelines.**KIEFormatBundle**(*img_to_float=True*)

Key information extraction formatting bundle.

Based on the DefaultFormatBundle, itt simplifies the pipeline of formatting common fields, including "img", "proposals", "gt_bboxes", "gt_labels", "gt_masks", "gt_semantic_seg", "relations" and "texts". These fields are formatted as follows.

- img: (1) transpose, (2) to tensor, (3) to DataContainer (stack=True)
- proposals: (1) to tensor, (2) to DataContainer
- gt_bboxes: (1) to tensor, (2) to DataContainer
- gt_bboxes_ignore: (1) to tensor, (2) to DataContainer
- gt_labels: (1) to tensor, (2) to DataContainer
- gt_masks: (1) to tensor, (2) to DataContainer (cpu_only=True)
- **gt_semantic_seg: (1) unsqueeze dim-0 (2) to tensor,**

  (3) to DataContainer (stack=True)
- relations: (1) scale, (2) to tensor, (3) to DataContainer
- texts: (1) to tensor, (2) to DataContainer

class mmocr.datasets.pipelines.**LoadImageFromNdarray**(*to_float32=False*, *color_type='color'*, *file_client_args={'backend': 'disk'}*)

Load an image from np.ndarray.

Similar with `LoadImageFromFile`, but the image read from `results['img']`, which is np.ndarray.

class mmocr.datasets.pipelines.**LoadTextAnnotations**(*with_bbox=True*, *with_label=True*, *with_mask=False*, *with_seg=False*, *poly2mask=True*, *use_img_shape=False*)

Load annotations for text detection.

参数

- **with_bbox** (*bool*) –Whether to parse and load the bbox annotation. Default: True.
- **with_label** (*bool*) –Whether to parse and load the label annotation. Default: True.
- **with_mask** (*bool*) –Whether to parse and load the mask annotation. Default: False.
- **with_seg** (*bool*) –Whether to parse and load the semantic segmentation annotation. Default: False.
- **poly2mask** (*bool*) –Whether to convert the instance masks from polygons to bitmaps. Default: True.
- **use_img_shape** (*bool*) –Use the shape of loaded image from previous pipeline `LoadImageFromFile` to generate mask.

process_polygons(*polygons*)

Convert polygons to list of ndarray and filter invalid polygons.

参数 **polygons** (*list[list]*) –Polygons of one instance.

返回 Processed polygons.

返回类型 list[numpy.ndarray]

class mmocr.datasets.pipelines.**MultiRotateAugOCR**(*transforms*, *rotate_degrees=None*, *force_rotate=False*)

Test-time augmentation with multiple rotations in the case that img_height > img_width.

An example configuration is as follows:

```
rotate_degrees=[0, 90, 270],
transforms=[
    dict(
        type='ResizeOCR',
        height=32,
        min_width=32,
        max_width=160,
        keep_aspect_ratio=True),
    dict(type='ToTensorOCR'),
    dict(type='NormalizeOCR', **img_norm_cfg),
    dict(
        type='Collect',
        keys=['img'],
        meta_keys=[
            'filename', 'ori_shape', 'img_shape', 'valid_ratio'
        ]),
]
```

After MultiRotateAugOCR with above configuration, the results are wrapped into lists of the same length as follows:

```
dict(
    img=[...],
    img_shape=[...]
    ...
)
```

参数

- **transforms** (*list[dict]*) –Transformation applied for each augmentation.
- **rotate_degrees** (*list[int] | None*) –Degrees of anti-clockwise rotation.
- **force_rotate** (*bool*) –If True, rotate image by ‘rotate_degrees’ while ignore image aspect ratio.

class mmocr.datasets.pipelines.**NerTransform**(*label_convertor*, *max_len*)

Convert text to ID and entity in ground truth to label ID. The masks and tokens are generated at the same time.

The four parameters will be used as input to the model.

参数

- **label_convertor** –Convert text to ID and entity
- **ground truth to label ID.** (*in*) –
- **max_len** (*int*) –Limited maximum input length.

class mmocr.datasets.pipelines.**NormalizeOCR**(*mean*, *std*)

Normalize a tensor image with mean and standard deviation.

class mmocr.datasets.pipelines.**OCRSegTargets**(*label_convertor=None*, *attn_shrink_ratio=0.5*, *seg_shrink_ratio=0.25*, *box_type='char_rects'*, *pad_val=255*)

Generate gt shrunk kernels for segmentation based OCR framework.

参数

- **label_convertor** (*dict*) –Dictionary to construct label_convertor to convert char to index.
- **attn_shrink_ratio** (*float*) –The area shrunk ratio between attention kernels and gt text masks.
- **seg_shrink_ratio** (*float*) –The area shrunk ratio between segmentation kernels and gt text masks.
- **box_type** (*str*) –Character box type, should be either ‘char_rects’ or ‘char_quads’, with ‘char_rects’ for rectangle with `xyxy` style and ‘char_quads’ for quadrangle with `x1y1x2y2x3y3x4y4` style.

generate_kernels(*resize_shape*, *pad_shape*, *char_boxes*, *char_inds*, *shrink_ratio=0.5*, *binary=True*)

Generate char instance kernels for one shrink ratio.

参数

- **resize_shape** (*tuple(int, int)*) –Image size (height, width) after resizing.
- **pad_shape** (*tuple(int, int)*) –Image size (height, width) after padding.
- **char_boxes** (*list[list[float]]*) –The list of char polygons.
- **char_inds** (*list[int]*) –List of char indexes.
- **shrink_ratio** (*float*) –The shrink ratio of kernel.
- **binary** (*bool*) –If True, return binary ndarray containing 0 & 1 only.

返回 The text kernel mask of (height, width).

返回类型 char_kernel (ndarray)

shrink_char_quad(*char_quad*, *shrink_ratio*)

Shrink char box in style of quadrangle.

参数

- **char_quad** (*list[float]*) –Char box with format [x1, y1, x2, y2, x3, y3, x4, y4].
- **shrink_ratio** (*float*) –The area shrunk ratio between gt kernels and gt text masks.

shrink_char_rect(*char_rect*, *shrink_ratio*)

Shrink char box in style of rectangle.

参数

- **char_rect** (*list[float]*) –Char box with format [x_min, y_min, x_max, y_max].
- **shrink_ratio** (*float*) –The area shrunk ratio between gt kernels and gt text masks.

class mmocr.datasets.pipelines.**OnlineCropOCR**(*box_keys=['x1', 'y1', 'x2', 'y2', 'x3', 'y3', 'x4', 'y4']*, *jitter_prob=0.5*, *max_jitter_ratio_x=0.05*, *max_jitter_ratio_y=0.02*)

Crop text areas from whole image with bounding box jitter. If no bbox is given, return directly.

参数

- **box_keys** (*list[str]*) –Keys in results which correspond to RoI bbox.
- **jitter_prob** (*float*) –The probability of box jitter.
- **max_jitter_ratio_x** (*float*) –Maximum horizontal jitter ratio relative to height.
- **max_jitter_ratio_y** (*float*) –Maximum vertical jitter ratio relative to height.

class mmocr.datasets.pipelines.**OpencvToPil**(***kwargs*)

Convert `numpy.ndarray` (bgr) to `PIL Image` (rgb).

class mmocr.datasets.pipelines.**PANetTargets**(*shrink_ratio=(1.0, 0.5)*, *max_shrink=20*)

Generate the ground truths for PANet: Efficient and Accurate Arbitrary- Shaped Text Detection with Pixel Aggregation Network.

[https://arxiv.org/abs/1908.05900]. This code is partially adapted from https://github.com/WenmuZhou/PAN.pytorch.

参数

- **shrink_ratio** (*tuple[float]*) –The ratios for shrinking text instances.
- **max_shrink** (*int*) –The maximum shrink distance.

generate_targets(*results*)

Generate the gt targets for PANet.

参数 **results** (*dict*) –The input result dictionary.

返回 The output result dictionary.

返回类型 results (dict)

class mmocr.datasets.pipelines.**PilToOpencv**(***kwargs*)

Convert `PIL Image` (rgb) to `numpy.ndarray` (bgr).

class mmocr.datasets.pipelines.**RandomCropInstances**(*target_size*, *instance_key*, *mask_type='inx0'*, *positive_sample_ratio=0.625*)

Randomly crop images and make sure to contain text instances.

参数

- **target_size** (*tuple or int*) –(height, width)
- **positive_sample_ratio** (*float*) –The probability of sampling regions that go through positive regions.

class mmocr.datasets.pipelines.**RandomCropPolyInstances**(*instance_key='gt_masks'*, *crop_ratio=0.625*, *min_side_ratio=0.4*)

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

sample_crop_box(*img_size*, *results*)

Generate crop box and make sure not to crop the polygon instances.

参数

- **img_size** (*tuple(int)*) –The image size (h, w).
- **results** (*dict*) –The results dict.

class mmocr.datasets.pipelines.**RandomPaddingOCR**(*max_ratio=None*, *box_type=None*)

Pad the given image on all sides, as well as modify the coordinates of character bounding box in image.

参数

- **max_ratio** (*list[int]*) –[left, top, right, bottom].
- **box_type** (*None|str*) –Character box type. If not none, should be either 'char_rects' or 'char_quads', with 'char_rects' for rectangle with `xyxy` style and 'char_quads' for quadrangle with `x1y1x2y2x3y3x4y4` style.

class mmocr.datasets.pipelines.**RandomRotateImageBox**(*min_angle=- 10*, *max_angle=10*, *box_type='char_quads'*)

Rotate augmentation for segmentation based text recognition.

参数

- **min_angle** (*int*) –Minimum rotation angle for image and box.
- **max_angle** (*int*) –Maximum rotation angle for image and box.

- **box_type** (*str*) –Character box type, should be either ‘char_rects’ or ‘char_quads’, with ‘char_rects’ for rectangle with `xyxy` style and ‘char_quads’ for quadrangle with `x1y1x2y2x3y3x4y4` style.

class mmocr.datasets.pipelines.**RandomRotateTextDet**(*rotate_ratio=1.0*, *max_angle=10*)

Randomly rotate images.

class mmocr.datasets.pipelines.**ResizeNoImg**(*img_scale*, *keep_ratio=True*)

Image resizing without img.

Used for KIE.

class mmocr.datasets.pipelines.**ResizeOCR**(*height*, *min_width=None*, *max_width=None*, *keep_aspect_ratio=True*, *img_pad_value=0*, *width_downsample_ratio=0.0625*, *backend=None*)

Image resizing and padding for OCR.

参数

- **height** (*int | tuple(int)*) –Image height after resizing.
- **min_width** (*none | int | tuple(int)*) –Image minimum width after resizing.
- **max_width** (*none | int | tuple(int)*) –Image maximum width after resizing.
- **keep_aspect_ratio** (*bool*) –Keep image aspect ratio if True during resizing, Otherwise resize to the size height * max_width.
- **img_pad_value** (*int*) –Scalar to fill padding area.
- **width_downsample_ratio** (*float*) –Downsample ratio in horizontal direction from input image to output feature.
- **backend** (*str | None*) –The image resize backend type. Options are *cv2*, *pillow*, *None*. If backend is None, the global imread_backend specified by `mmcv.use_backend()` will be used. Default: None.

class mmocr.datasets.pipelines.**ScaleAspectJitter**(*img_scale=None*, *multiscale_mode='range'*, *ratio_range=None*, *keep_ratio=False*, *resize_type='around_min_img_scale'*, *aspect_ratio_range=None*, *long_size_bound=None*, *short_size_bound=None*, *scale_range=None*)

Resize image and segmentation mask encoded by coordinates.

Allowed resize types are *around_min_img_scale*, *long_short_bound*, and *indep_sample_in_range*.

class mmocr.datasets.pipelines.**TextSnakeTargets**(*orientation_thr=2.0*, *resample_step=4.0*, *center_region_shrink_ratio=0.3*)

Generate the ground truth targets of TextSnake: TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes.

[https://arxiv.org/abs/1807.01544]. This was partially adapted from https://github.com/princewang1994/TextSnake.pytorch.

参数 **orientation_thr** (*float*) –The threshold for distinguishing between head edge and tail edge among the horizontal and vertical edges of a quadrangle.

draw_center_region_maps(*top_line*, *bot_line*, *center_line*, *center_region_mask*, *radius_map*, *sin_map*, *cos_map*, *region_shrink_ratio*)

Draw attributes on text center region.

参数

- **top_line** (*ndarray*) –The points composing top curved sideline of text polygon.
- **bot_line** (*ndarray*) –The points composing bottom curved sideline of text polygon.
- **center_line** (*ndarray*) –The points composing the center line of text instance.
- **center_region_mask** (*ndarray*) –The text center region mask.
- **radius_map** (*ndarray*) –The map where the distance from point to sidelines will be drawn on for each pixel in text center region.
- **sin_map** (*ndarray*) –The map where vector_sin(theta) will be drawn on text center regions. Theta is the angle between tangent line and vector (1, 0).
- **cos_map** (*ndarray*) –The map where vector_cos(theta) will be drawn on text center regions. Theta is the angle between tangent line and vector (1, 0).
- **region_shrink_ratio** (*float*) –The shrink ratio of text center.

find_head_tail(*points*, *orientation_thr*)

Find the head edge and tail edge of a text polygon.

参数

- **points** (*ndarray*) –The points composing a text polygon.
- **orientation_thr** (*float*) –The threshold for distinguishing between head edge and tail edge among the horizontal and vertical edges of a quadrangle.

返回 The indexes of two points composing head edge. tail_inds (list): The indexes of two points composing tail edge.

返回类型 head_inds (list)

generate_center_mask_attrib_maps(*img_size*, *text_polys*)

Generate text center region mask and geometric attribute maps.

参数

- **img_size** (*tuple*) –The image size of (height, width).
- **text_polys** (*list[list[ndarray]]*) –The list of text polygons.

返回

The text center region mask. radius_map (ndarray): The distance map from each pixel in text

center region to top sideline.

sin_map (ndarray): The sin(theta) map where theta is the angle between vector (top point - bottom point) and vector (1, 0).

cos_map (ndarray): The cos(theta) map where theta is the angle between vector (top point - bottom point) and vector (1, 0).

返回类型 center_region_mask (ndarray)

generate_targets(*results*)

Generate the gt targets for TextSnake.

参数 **results** (*dict*) –The input result dictionary.

返回 The output result dictionary.

返回类型 results (dict)

generate_text_region_mask(*img_size*, *text_polys*)

Generate text center region mask and geometry attribute maps.

参数

- **img_size** (*tuple*) –The image size (height, width).
- **text_polys** (*list[list[ndarray]]*) –The list of text polygons.

返回 The text region mask.

返回类型 text_region_mask (ndarray)

reorder_poly_edge(*points*)

Get the respective points composing head edge, tail edge, top sideline and bottom sideline.

参数 **points** (*ndarray*) –The points composing a text polygon.

返回

The two points composing the head edge of text polygon.

tail_edge (ndarray): The two points composing the tail edge of text polygon.

top_sideline (ndarray): The points composing top curved sideline of text polygon.

bot_sideline (ndarray): The points composing bottom curved sideline of text polygon.

返回类型 head_edge (ndarray)

resample_line(*line*, *n*)

Resample n points on a line.

参数

- **line** (*ndarray*) –The points composing a line.
- **n** (*int*) –The resampled points number.

返回 The points composing the resampled line.

返回类型 resampled_line (ndarray)

resample_sidelines(*sideline1*, *sideline2*, *resample_step*)

Resample two sidelines to be of the same points number according to step size.

参数

- **sideline1** (*ndarray*) –The points composing a sideline of a text polygon.
- **sideline2** (*ndarray*) –The points composing another sideline of a text polygon.
- **resample_step** (*float*) –The resampled step size.

返回 The resampled line 1. resampled_line2 (ndarray): The resampled line 2.

返回类型 resampled_line1 (ndarray)

class mmocr.datasets.pipelines.**ToTensorNER**

Convert data with `list` type to tensor.

class mmocr.datasets.pipelines.**ToTensorOCR**

Convert a `PIL Image` or `numpy.ndarray` to tensor.

mmocr.datasets.pipelines.**sort_vertex**(*points_x*, *points_y*)

Sort box vertices in clockwise order from left-top first.

参数

- **points_x** (*list[float]*) –x of four vertices.
- **points_y** (*list[float]*) –y of four vertices.

返回 x of sorted four vertices. sorted_points_y (list[float]): y of sorted four vertices.

返回类型 sorted_points_x (list[float])

mmocr.datasets.pipelines.**sort_vertex8**(*points*)

Sort vertex with 8 points [x1 y1 x2 y2 x3 y3 x4 y4]

25.3 utils

class mmocr.datasets.utils.**HardDiskLoader**(*ann_file*, *parser*, *repeat=1*)

Load annotation file from hard disk to RAM.

参数 **ann_file** (*str*) –Annotation file path.

class mmocr.datasets.utils.**LineJsonParser**(*keys=[]*)

Parse json-string of one line in annotation file to dict format.

参数 **keys** (*list[str]*) –Keys in both json-string and result dict.

class mmocr.datasets.utils.**LineStrParser**(*keys=['filename', 'text']*, *keys_idx=[0, 1]*, *separator=' '*, ***kwargs*)

Parse string of one line in annotation file to dict format.

参数

- **keys** (*list[str]*) –Keys in result dict.
- **keys_idx** (*list[int]*) –Value index in sub-string list for each key above.
- **separator** (*str*) –Separator to separate string to list of sub-string.

class mmocr.datasets.utils.**LmdbLoader**(*ann_file*, *parser*, *repeat=1*)

Load annotation file with lmdb storage backend.

CHAPTER 26

导引

- genindex
- search

Python 模块索引

m

索引

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

模块