NNDL Project3

19307130195 Lesi Chen

Abstract

I finish the **Scene Text Recogition** task with a second-stage model, to be more extact, I use **PSENet** for the text detection part and **Robust Scanner** for the text recogition part.

Without any pretrained backbone or pretrained model, without any extra datasets, I seperately trained the detector and the recognizer. To be more exact, I trained the PSENet for **160** epoches and it got the F1 Score of **0.57**, with setting of the miou threshold is 0.5. And, I trained the Robust Scanner for **200** epoches and it finnaly got a word accuracy of **0.58** and normlized edit distance metric(1-N.E.D.) of **0.82**. After training the two stages seperately, I merge the two stage to get an end-to-end model, and I make an evaluation on the validation set(which I split out from the traning set before training), and the end-to-end model got a F1 score of **0.35** (the score defined by the evaluation protocol of this project)

Deeply thankful to mmlab and their open source project for OCR tasks:**mmocr**, this project is heavily base on mmocr.

My codes are also published in my github: https://github.com/TrueNobility303/scene-text-recognition

PSENet

Motivation

In my opinion,there exits some challenges in the task of scene text detection.

- curve texts
- multiscale texts
- squeezed texts

Many models are based on object detection models, they can be defined as regression-based methods, such as Faster-RCNN and SSD, the regression-based methods also include the models which modified the vallina objection detection model by adding support for rotation angles(rotation proposals in RRPN, ROI rotate operation in FOTS,etc.) In my opinion, these models is with high probability to fail in scene text detection task, especially in the task of this project. The regression-base methods may remedy their defect by training on a large dataset generated by algorithms, such as the famous synthetic text

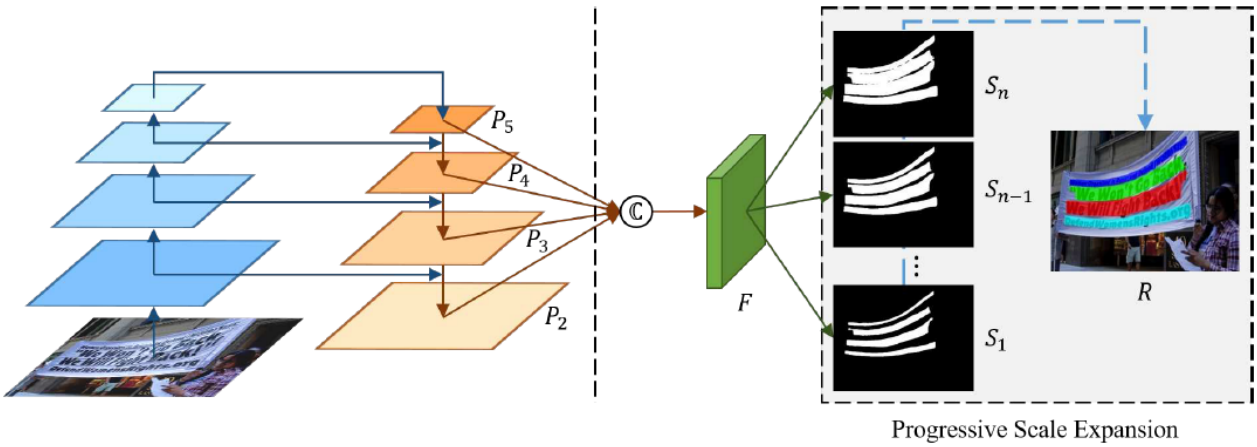
images dataset SynthText. However, I beleive that in this project, with only a relativalt quite smaller training set, where curved texts are widely distributed, the regression-based methods is likelt to fall short when handling this challenge.

Considering the **curved text challenge** mentioned above, I adpot the segmentation-based methods for scene text detection in this project, which are totally different from the regression-based methods. The segmention-based methods are based on the models which are designed for semantic segmentation or instance segmentation. By utilizing the segmention-based methods, it is possible for the network to detect arbitrary-shaped text instances . One representative method is PSENet(Progressive Scale Expansion Network). Besides of handling the cuved texts well, PSENet use a progressive scale expansion structure to handle the **multiscale texts** and **squeeze texts** challenges. That is to say, the texts in real-world scene are of multi scales and many of them may have a close distance to the nearby text instances. Many segmentation methods benifit from the structrue of FPN(Feature Pyramic Network) to handle the mutilscale chanllenges, which is also an important structure in PSENet.However, the squeezed text chanlenge is difficult for naive semantic segmentation methods to deal with, because naive semantic segmentation methods are likely to label the nearby text instances the same, which may lead to many failures in scene text detection.

PSENet handle the above challenges well, and this is why I use PSENet in this project.

Network Structure

PSENet generates the different scale of kernelsfor 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, PSENet is effetive to split the close text instances, making it easier to use segmentation-based methods to detect arbitrary-shaped text instances. From my perspective, PSENet is a great model, which not only avoid the fatal defect in regression-based methods, but also modified the vallina segmentation-based methods to garantee a state-of-the-art performance.



Overview of PSENet

The overview of PSENet is shown in the above picture.The left part of pipeline is implemented from FPN.The right part denotes the feature fusion and the progressive scale expansion algorithm.

FPN fuse the features of different scales to get the feature map with upsample and concatenation function.With the feature map, PSENet first generated minimal kernel's map by finding connected components. After that, PSENet use a procedure of progressive scale expansion algorithm, which is based on BFS(Breadth-First-Search) to generate segmentation results of different shrink scale ratios. After steps of progressive scale expansion, PSENet generate the complete text instance. When training, PSENet also shrink the ground truth text instance by Vatti clipping algorithm to attain multi-scale mask labels for learning.

Loss Function

The loss of PSENet(PSELoss) L consists of the losses for the complete text instances $L_c$and the losses of shrunk text instancesL_s.

$$L = \lambda L_c + (1 - \lambda)L_s$$

When calculating the losses, PSELoss adopt dice coefficient and OHEMLoss(Online Hard Example Mining). For example, the losses of the complete text instances can be formulated as:

$$L_c = 1 - D(P * M, G * M)$$

where D refers to the dice coefficient, G the ground truth and P the prediction, and M is a mask of hard exmaple with a threshold value of 0.5.

Traning Strategy

I use the following data argumentation strategies.

- horizonal random flip
- random crop
- random rotate

As for optimizer, I utilize SGD with the learning rate of 0.08, momemtum of 0.9, weight decay of 1e-4.

As for scheduler, I utilize linear warmup with a warm-up iterations of 500 and warm-up ratio of 1e-3. I trained the PSENet by 160 epoches, and I also use the step-learning rate-scheduler, with which I decay the laerning rate at epoch 80 and epoch 128.

To make sure the network to train from scratch, I set pretrained=False in the backbone.

Result

Setting the miou threshhold=0.5,we get the following result of PSENet.

precision	**recall**	**hmean**
0.6979	0.4884	0.5746

Here shows a demo,we can see that the detector works quite well,except few small flaws.

Demo of text detecttion

Robust Scanner

Motivation

Traditional scene text recognition methods,could be roughly classified into two branches: the recurrent neural network (RNN) based recognizers and the convolutional neural network (CNN) based ones. CRNN combine CNN and RNN to learn spatial dependencies
and applies CTC to translate per-slice prediction into a label sequence. However, as proved repeatedly in recent years, the self-attention based methods(or transformer based methods) have more expreesive power than either CNN or RNN. Some recent method like NRTR uses solely self-attention mechanism (as in the Trasnfomer) as the fundamental module, and defeated many CNN-based or RNN-based model. Therefore, in this project, I focus on the **self-attetion** based methods to get a better result.

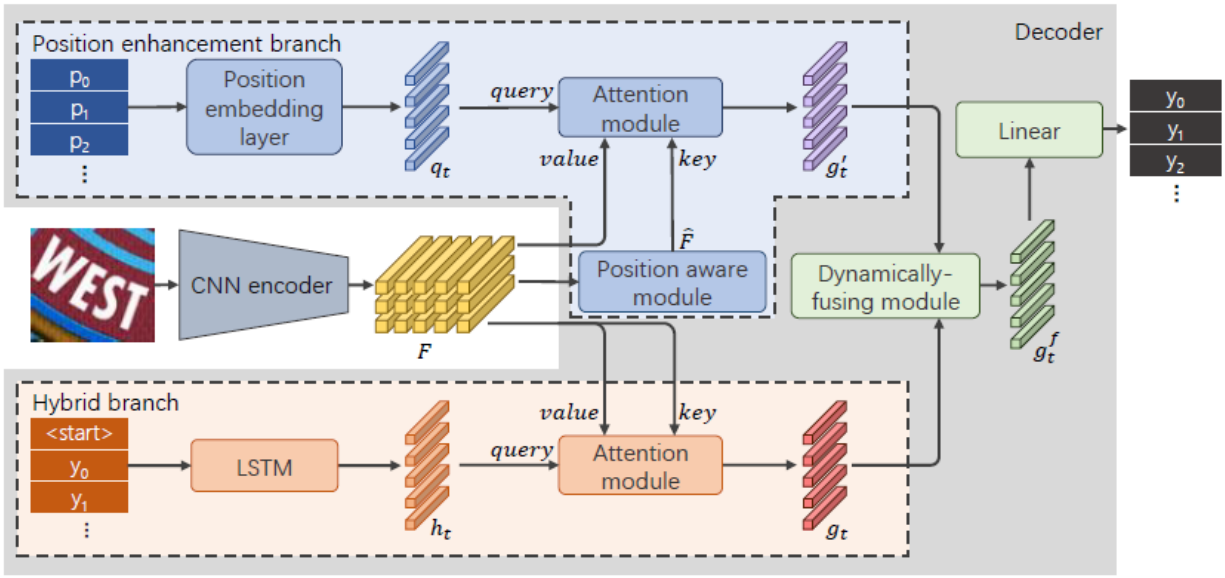
Self-attention based method have both the advantages of CNN and thos eof RNN. To be more exact, self-attention methods have a global receptive field instead of the local receptive field in CNN-based method. In other words, self-attention methods enable networks to learn dependencies among distant positions ,which are difficult for CNN to learn.Plus, self-attention mechanisms enable to compute hidden representation in parallel, which is impossible for RNN-based method, and self-attention methods also avoid the gradient vanishing/exploding problems in RNN.

However, since self-attention module contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, it is a must to inject some information about the relative or absolute position of the tokens in a sequence. The original transfomer attempt to solve this problem by adding "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. Despite of these, most attention based frameworks contain the hybrid information of context and position. The positional clues become weaker while the contextual ones become stronger as the time step increases during decoding, which may lead to attention drift and misrecognition, especially on contextless text images.

Robust Scanner mitigate the issue of misrecogniton in con- textless scenarios via introducing a novel **position enhancement branch** and a **dynamic fusion module**. Plus, Robust Scanner is a hybrid model which mix covolution, recurrent neural cell(LSTM), and self-attention to benefit from every part of the whole network. Therefore, I regard Robust Scanner as a successful example in text recognition task, and I choose Robust Scanner to finish my second stage of scene text recognition.

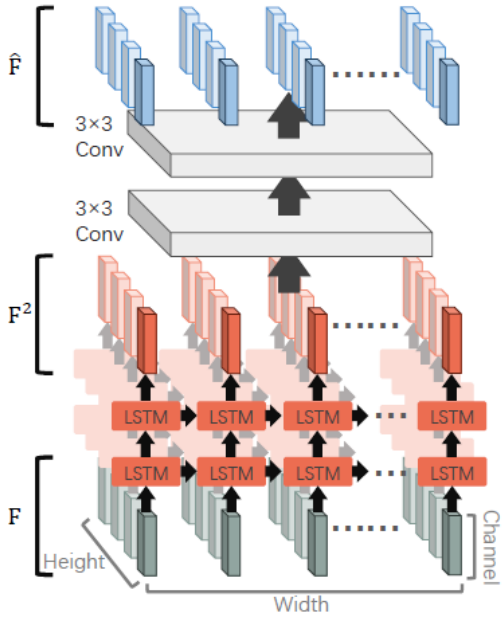
Network Structure

The overview of Robust Scanner is shown in the picture.



Overview of Robust Scanner

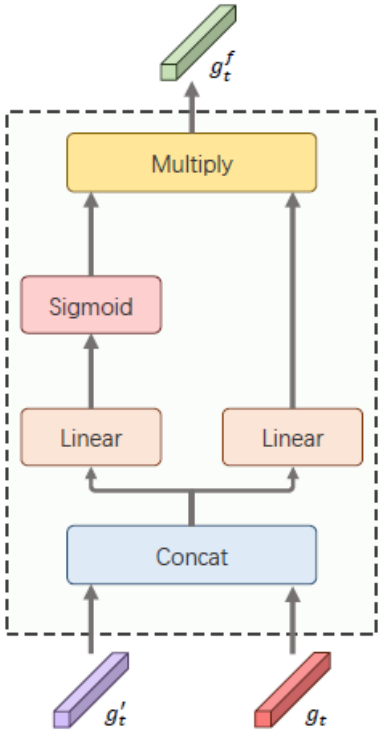
The CNN encoder is like the CNN encoder in most methods such as CRNN. The Hybrid branch utilizes both contextual and positional information simultaneously, as in many self-attention based method, consisting of one two-layer and one attention module. The novel structures in Robust Scanner is the position enhancement, consisting of one position embedding layer, one position aware module and one attention module.



Position Aware Module

The attention module in position enhancement branch is also modified to a position aware module. The position aware module is illustrated in the above picture. It utilizes two LSTM modules to capture the global context, with the help of the position information. Plus, the position aware module uses covolution modules to capture high-level information.

Getting the glimpse vector g_t' containing precious position infomation, and the contextual vector g_t containing the hybrid infomation. Robust Scanner uses a Dynamically-fusing module to fuse the two infomation, which is shown in the picture.



Dynamically-fusing module

Training Staregy

As in the original paper of Robust Scanner, the RobustScanner in this project is trained from scratch using Adam optimizer with
the base learning rate 1e-3.

I trained the network for 200 epoches whiout step scheduler.

I trained without any extra data argumentation.

Result

I measure the result with the both word-level measures and character-level measures. The details are in the following tables.

Here shows the character-level result:

character precision	**character recall**
0.7908	0.7752

And the word-level result:

word accuracy	**normalized edit distance metric**
0.5822	0.8189

with the following fomular:

$$\text{normalized edit distance metric} = 1 - \frac{1}{N}\sum_{i=1}^{N} D(p_i, g_i)/\max(p_i, g_i)$$

where the D is defined as edit distance of the prediction p_i and the gound truth g_i,

Here we can see a correct predition generated by the text recognition network, the word in the image is SUGGESTIONS and the network recognised it. The first line is the predition, and the second line is the image for recognition.

Demo of text recogintion

Final Model

I merge PSENet and Robust Scanner to get the final end-to-end model.And a demo is below,we can see in the picture that all the predition generated by the end-to-end model is correct. (The left picture shows the mask prediction given by PSENet, the right picture shows the bounding boxes and the text prediction of each bounding box)

Demo of end-to-end result

We defined the measure as the evaluation protocol of this project, to be more exact, a predition p is regared as a posivite sample when it satisfies the following formula:

$$M(A(p), A(g)) > 0.5 \ \wedge \ D(t(p), t(g)) = 0$$

where M measues the moiu, D measures the edit distance of a transcript, A for the Area,t for the transcript,g for ground truth.

With the above definition, the results of the end-to-end model are in the table:

precision	**recall**	**f1 score**
0.359	0.339	0.349

Extra Details

- I investigated all the text recogintion models and text detection models in the model zoo of mmocr, and select the most suitable models from my point of view according to the reasons I mentioned in the motivation part. Specifically, I choose a representative text detection model PSENet with: segmentation-based method, Feature Pyramic Network, other important techniques for solving squeezed texts challenges, and a representative text recognition model Robust Scanner with: encoder-decoder based model, self-attention mechanism, combination of CNN,RNN and Transformer modules.
- As the mmocr framework does not support other languages except English and Chinese, I modified the scource code of mmocr to offer support to multi languages, such as the language in this project, Hindin, Arabic, Korean, etc.
- Unlike the original PSENet which uses the ResNet pretrained on ImageNet, I trained PSENet without ImageNet pretrained parameters. Also, due to the limitation of hardware and time, I merely train for 160 epoches instead of 600 epoches in the experiment setup in the original paper of PSENet. Therefore, I adjusted the training strategy, which is mentioned in the stratagy part.

  Plus, the original Robust Scanner was trained on many large datasets, including sythetic datasets such as SynthText and MJSynth. Instead, in this project, only a small scene text dataset is provided. Therefore, I also adjusted the training strategy with 160 epoches.
- Since no validation set is provided, I split the traning set with the ratio 9:1 to get a validation set to evaluate the performance of the models.

Getting Started

The mmocr framework have implemented the models for both text detection and text recognition. However, in this project, we are dealing with an end-to-end scene text recoginition task, instead of two seperate tasks. I modified the scource codes of mmocr, implementing a pipeline for scene text recognition task. And it is easy to get strated according to the instructions.

Data Preparation

To get a detection dataset from icdar format data for training a text detector, you can run

```
python -u icdar2detset.py
```

The above program also split the traning set.

To get a recognition dataset from the detection dataset, you can run

```
python -u detset2recset.py
```

Train the networks

After preparing dataset for training, you can train a network by running

```
./mmocr/tools/dist_test.sh ${CONFIG_FILE} ${CHECKPOINT_FILE} ${GPU_NUM} [--optional]
```

For example, you can reproduce my results of PSENet by running

```
nohup ./mmocr/tools/dist_test.sh mmocr/configs/textdet/psenet/my_psenet.py psenet 1 --gpu-ids 0
```

Or my results of Robust Scanner

```
nohup ./mmocr/tools/dist_test.sh mmocr/configs/textrecog/robust_scanner/my_robust_scanner.py
rbsan 1
```

Test the end-to-end model

After the two stages of training, you can also gernerate the end-to-end output given by the detector and recognizer

```
python -u mmocr/demo/gernerate_output.py
```

And evaluate with the validation set

```
python -u mmocr/demo/measure.py
```

Also, if you are insteretd in the prediction of certain image, please run

```
python -u demo/ocr_image_demo.py ${INPUT_IMAGE} ${OUTPUT_IMAGE}
```

For example, you can get the demo result which I have shown above, by runing

```
python -u demo/ocr_image_demo.py detset/imgs/test/img_93.jpg results/output.png
```

If you are interested in the result of only the detector or recognizer, run,

```
python -u demo/image_demo.py ${TEST_IMG} ${CONFIG_FILE} ${CHECKPOINT_FILE} ${SAVE_PATH}
```

Predition Summited

The summied predition given by my model can be found in /myicdar/testgt.

Here I show one of the result, with the first image of test set. I output all the prection scripts label Latin, because it will be ignored when evaluation.

Input image

And the prediction in my summit

```
293,93,414,87,416,128,295,134,Latin,الجمال
411,97,500,97,500,132,411,132,Latin,عزب
289,188,293,136,490,149,487,200,Latin,CamelPens
291,327,292,292,424,297,423,332,Latin,KHANSAHEB
```

If you check or visualize the prediction, you will find it totally correct! And that finish my report.

REF

[1] Shape Robust Text Detection with Progressive Scale Expansion Network

[2] RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition

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

[4] ICDAR2019 Robust Reading Challenge on Arbitrary-Shaped Text

[5] https://github.com/open-mmlab/mmocr