Multi-lingual Scene Text Detection and Recognition

Song Pengwei 17307110440 **Zhang Yuqing 16307130308**
School of Data Science, Fudan University

Abstract

1 INTRODUCTION

Reading scene text in natural scene images is a key component in diverse applications,such as data mining of street-view-like images for information used in geographic information systems. Scene text detection and recognition finds its use in larger integrated systems such as those for autonomous driving, indoor navigation and visual search engines.

In this project, the objective is to localize a set of bounding boxes and their corresponding transcriptions given an input scene image –in other words, a STR task. In this task, we used 7932 images for training the network, and used 1990 images for evaluation. The natural scene images containing text instances from `ARABIC`, `BANGLA`, `CHINESE`, `DEVANAGARI`, `ENGLISH`, `FRENCH`, `GERMAN`, `ITALIAN`, `JAPANESE` and `KOREAN`.

To achieve the goal, normally the method is to split the task into two sub-tasks:
Text Detection: detect or segment the area where text may appear,
Text Recognition: recoginize the text information in the detected or segmented area, namely convert the image infomation to the text infomation.

Figure 1: the process of STR

This paper is organized as follows. Firstly, we introduced some related algorithms of the STR task. Then we focus on illustrating our algorithms and experimental results . And in the last section, we will discuss our thoughts on improving the detection and recognition performance to promote future research.

2 RELATED WORK

2.1 Detection

For general object detection, researchers have designed some powerful models, like SSD, YOLO, Faster R-CNN. However, directly using them on the scene text detection often result in a unexpected performance. The major reason is that scene text is often a fine-grained object with varying line length and ratio of length to width.
There have been many efforts to tackle the detection task. CTPN regard the text line as a character

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

sequence rather than a single independent target in object detection task. Text characters of each line are mutually context, which can be utilized in the training process to improve the performance. RRPN introduces the rotation factor in the bounding box. In this framework, the ground truth of a text area is expressed as a rotating five tuples $(x; y; h; w; \theta)$ where θ is omitted in the traditional object detection models.

2.2 Recognition

Nowadays, there are two main technologies of end-to-end OCR based on deep learning: CRNN OCR and attention OCR. In fact, the main difference between the two methods lies in the final output layer (translation layer), that is, how to transform the sequence feature information learned by the network into the final recognition result. In the feature learning phase, the two main technologies adopt the network structure of CNN + RNN. The alignment method of CRNN OCR is CTC algorithm, while the attention OCR is the attention mechanism.

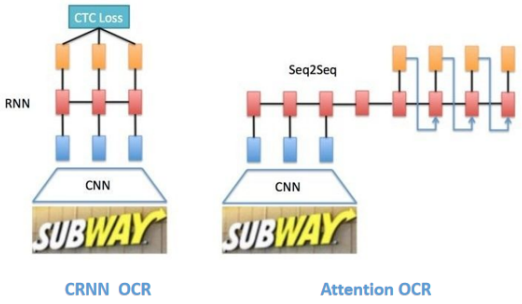


Figure 2: CRNN OCR and attention OCR

In our project, we also use CRNN for recognition. It contains CNN feature extraction layer and BLSTM sequence feature extraction layer, which can carry out end-to-end joint training. It uses BLSTM and CTC components to learn the contextual relationship within characters, thereby effectively improving the accuracy of text recognition and making the model more robust. In the prediction process, the front end uses a standard CNN network to extract the features of the text image, and uses BLSTM to fuse the feature vectors to extract the context features of the character sequence, and then obtain the probability distribution of each column of features. Finally, the text sequence is obtained through prediction by the transcription layer.

3 ALGORITHMS

In our project, we use CTPN for detection and CRNN for recognition.

3.1 Detection–CTPN

In our project, we use the Connectionist Text Proposal Network that directly localizes text sequences in convolutional layers in the detection stage. It combines CNN and LSTM deep network and it can detect the horizontally distributed text for complex scenes effectively.

3.1.1 The proposal and innovation of CTPN

CTPN algorithm is proposed for the following reasons:
(1) assuming that the text is horizontal;
(2) the text can be regarded as composed of every "letter". The letters here can be thought of as "fragments". The reason for this idea is that the algorithm based on general target detection is difficult to be adapted to STR, such as the text in the image whose length varies greatly. Therefore, the author decouples the text in the horizontal direction and divides it into each small piece, then

transforms the detection of the text line into the detection of the small piece, and finally uses the rules to combine the small pieces belonging to the same horizontal line.

The innovation of CTPN mainly consists of the following three points:

(1) split the text line into slice for detection, so that only the height of the text needs to be set prior anchor in the detection process.

(2) The author believes that the text is sequential, that is consistent with reading habits, from left to right. So the author adds RNN to get the semantics.

(3) post processing algorithm: text connection algorithm.

The structure difference of CTPN and RPN can be seen that: CTPN is the updating version of RPN which uses BLSTM to obtain information of timing direction. And CTPN uses horizontal slice box as regression target.

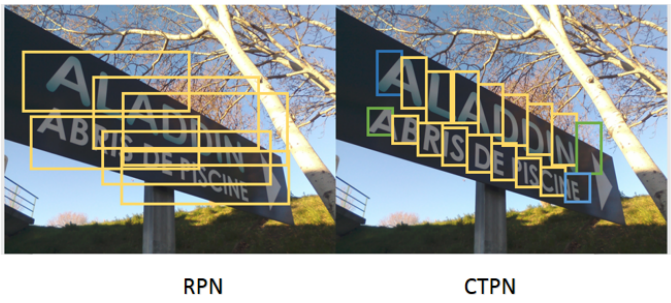


Figure 3: result difference of CTPN and RPN

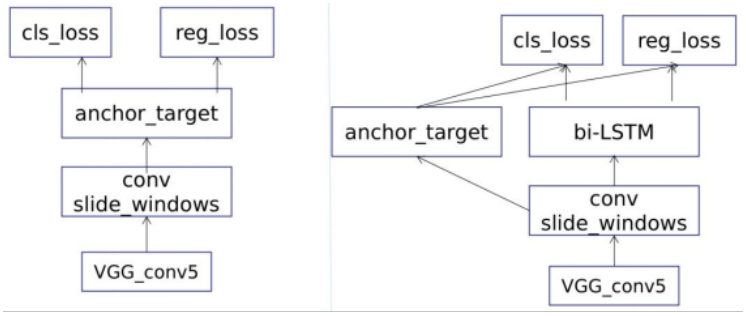


Figure 4: structure difference of CTPN and RPN

3.1.2 The structure of CTPN

1. Firstly, the feature is extracted through the BackBone architecture network **VGG16**, whose conv5 layer outputs the feature map of N x C x H x W.. One pixel in the feature map output by conv5 layer corresponds to 16 pixels of the original image because the accumulated stride of the convolution network of vgg16 through four pooling layers is 16.

2. Then make a 3 x 3 sliding window on conv5, that is, each point combines the features of the surrounding 3 x 3 region to obtain a feature vector with a length of 3 x 3 x C. As shown in the figure

below, the output is a feature map of N x 9C x H x W, which is still a spatial feature learned by CNN.

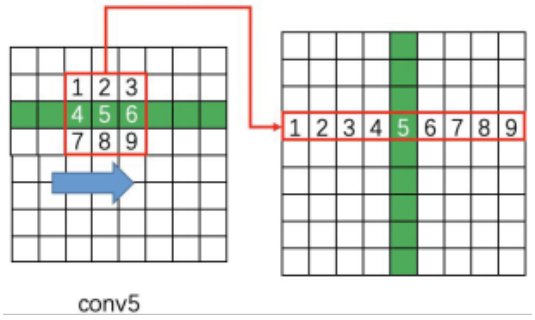


Figure 5: feature map of conv5 in CTPN

3. Reshape the feature map N x 9C x H x W to (NH) x W x 9C.
4. Input the data flow with batch = NH and the maximum time length Tmax = w into BLSTM to learn the sequence characteristics of each line. Then reshape (NH) x W x 256 to N x 256 x H x W.This feature contains not only spatial features, but also sequence features learned by BLSTM.
5. After passing through the FC layer, it becomes N x 512 x H x W.
6. Finally, through the RPN network similar to fast RCNN, text proposals are obtained. There are two brankes in RPN:
The branch below is used for bounding box regression. Because each point of FC feature map is equipped with 10 anchors, and only two values (center coordinate and height) are regressed, so RPN_ bbox_pred has 20 channels.

The branch above is used for softmax classification anchor.

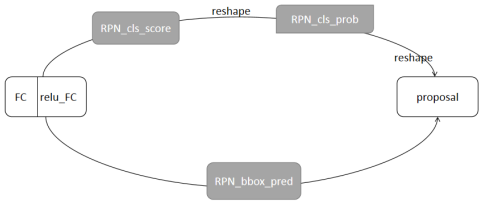


Figure 6: two branches in RPN to produce text proposals

3.1.3 Vertical anchor

CTPN aims at the text detection of horizontal arrangement, so it uses a group of (10) equal width anchors to locate the text position. Anchor width and height are:

$$\text{widths} = [16]$$

$$\text{heights} = [11, 16, 23, 33, 48, 68, 97, 139, 198, 283]$$

The setting of anchor is to ensure that in the X direction, anchor covers every point of the original image and does not overlap each other.Additionally, the height difference of different texts in Y direction is very large. So the height of anchors is set to 11-283, which is used to cover the text targets of different heights. CTPN equips 10 anchors for each point on FC Feature map.

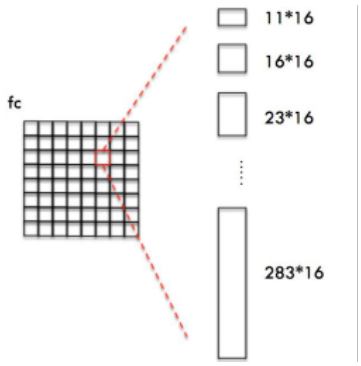


Figure 7: 10 anchors for each point on FC Feature map

After acquiring anchor, similar to faster RCNN, CTPN will do the following:

1. Softmax judges whether there is text in anchor, that is to say, select a positive anchor with a softmax score;

2. Bounding box expression corrects the center coordinate and height of anchor containing text:

$$\begin{aligned} v_c &= (c_y - c_y^a)/h^a, \qquad v_h = \log(h/h^a) \\ v_c^* &= (c_y^* - c_y^a)/h^a, \qquad v_h^* = \log(h^*/h^a) \end{aligned}$$

where $v = (v_c, v_h)$ is the coordinate predicted by regression; $v = (v_c^*, v_h^*)$ is the Ground Truth; c_y^a and h^a are the center coordinate and height of anchor respectively.

After the above softmax and direction binding box region processing, anchor will get a set of vertical bar like text proposals as shown in the figure below. The text position can be obtained by connecting these text proposals with the text line construction algorithm.

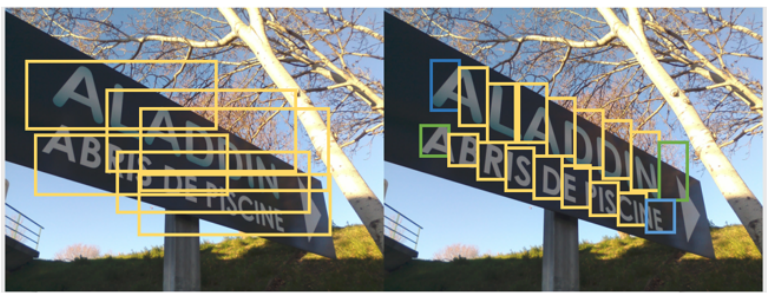


Figure 8: text proposals produced by RPN and CTPN

3.1.4 text line construction algorithm

By using the method of text line construction, these text proposals are connected into a text detection box.

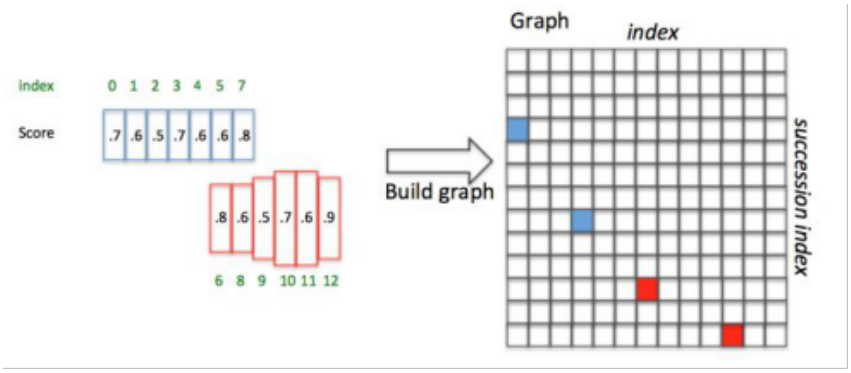


Figure 9: text detection box

In order to illustrate the problem, suppose a graph has two text proposals shown in the figure above, namely two groups of anchors in blue and red. CTPN uses the following algorithm to construct text lines:
Suppose that each anchor index is the green number, and each anchor softmax score is like a black number. The text line construction algorithm establishes the pair (Box_i , Box_j) of each anchor Box_i as follows:
Forward looking:
1. Along the horizontal direction, find the candidate anchors whose horizontal distance from Box_i is less than 50;
2. From the candidate anchors, select the anchor with the overlap of Box_i's vertical direction > 0.7;
3. Select the Box_j with the largest softmax score
Reverse search:
1. Along the horizontal negative direction, find the candidate anchor whose horizontal distance from Box_j is less than 50;
2. From the candidate anchors, select the anchorwith the overlap of Box_j's vertical direction > 0.7;
3. Pick out the Box_k with the largest softmax score.
Finally, compare $score_i$ and $score_k$:
1. If $score_i$ >= $score_k$, then this is the longest connection, then set $Graph(i, j) = True$;
2. If $score_i$ < $score_k$, it means that the connection must be included in another longer connection.

For example, as shown in the figure above, anchor has been arranged in x order, and has softmax score in the figure (the score here is given casually, only used to explain the construction algorithm of text line):
For the box3 with $I = 3$, look forward for 50 pixels, and the box7 with the largest score is $j = 7$; the box7 reverse search, and the box3 with the largest score is $k = 3$. Since score3 >= score3, pair (box3, box7) is the longest connection, set $graph(3, 7) = true$.
For box4, look forward to get box7; for box7, look backward to get box3, but score4 < score3, that is, pair (Box4, box3) is not the longest connection, which is included in pair (box3, box7).
Then, a connect graph of N x n (where n is the number of positive anchors) is established. Traverse graph:
1. Graph (0,3) = true and graph (3,7) = true, so anchor index 1 -> 3 ->7 forms a text, that is, a blue text area.
2. Graph (6,10) = true and graph (10,12) = true, so anchor index 6 -> 10 -> 12 forms another text, namely the red text area.
This determines the text detection box through text proposals.

3.1.5 CTPN training strategy

There are three parts in loss function:

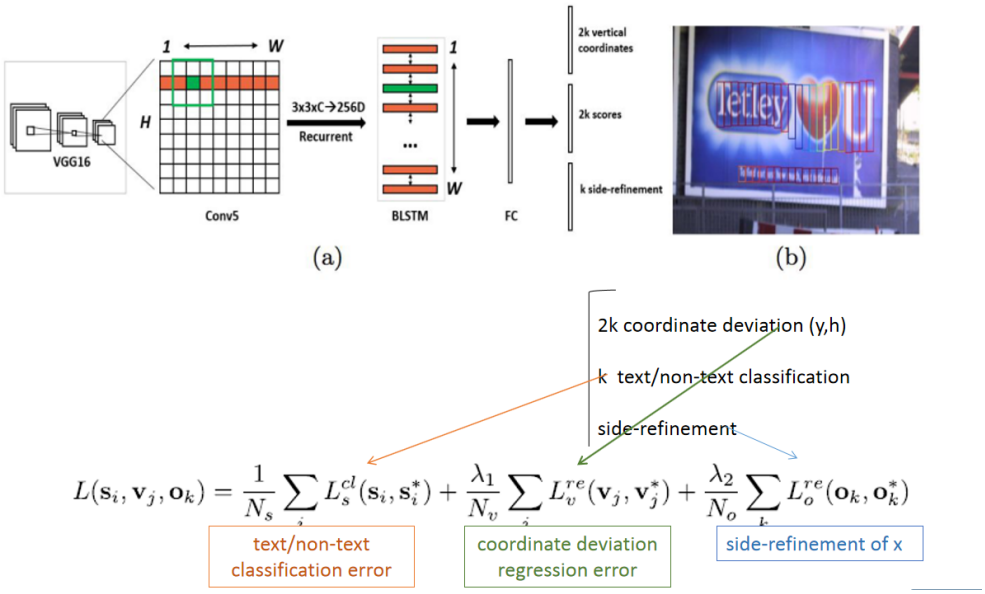


Figure 10: loss function of CTPN

Anchor foreground and background classification error: this loss is used to monitor whether text is included in each anchor. $s_i^* = 0, 1$ indicates whether it is a groove truth.
Vertical coordinate offset regression error: this loss is used to supervise the learning of the bearing box region Y direction offset of each anchor, similar to smooth L1 loss. Where VJ is the anchor with text in Si, or with groud truth vertical IOU > 0.5.
Correction error of X at the boundary: the loss is used to supervise the learning of offset in the direction X of bounding box expression of each anchor containing text, which is the same as that in the direction Y.
Note that in the training process of bounding box expression, you only need to pay attention to the positive anchor, and you don't need to care about the messy negative anchor. This is similar to faster R-CNN.

3.1.6 summary of CTPN

1. Due to the addition of LSTM, CTPN has a very good effect on horizontal text detection.
2. For the reason of anchor setting, CTPN can only detect horizontally distributed text, and we can detect vertical text by adding horizontal anchor. However, due to the limitation of the framework, the detection effect of irregular text is not good as expected.

3.2 Recognition–CRNN

3.2.1 Introduction of CRNN

Convolutional Recurrent Neural Network, the full name of CRNN, is used for end-to-end recognition of indefinite length of text sequence. Instead of cutting single text first, CRNN transforms text recognition into sequence learning problem based on time sequence, that is, image-based sequence recognition.

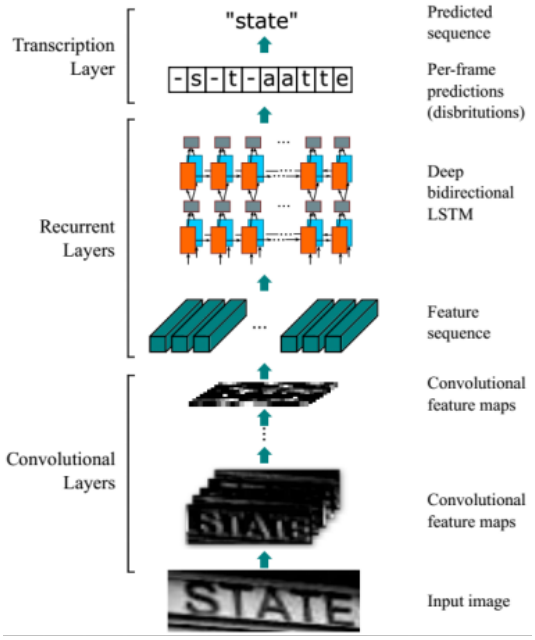


Figure 11: main structure of CRNN

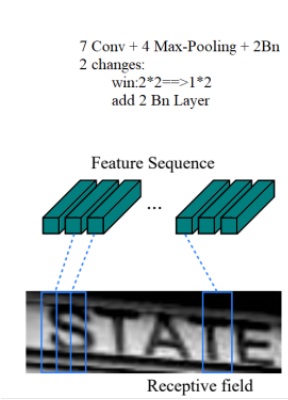
The whole CRNN network structure consists of three parts, from bottom to top:

1. CNN (convolution layers): use deep CNN to extract features from the input image to get the feature map;

2. RNN (recurrent layer): use bi-RNN (BLSTM) to predict the feature sequence, learn each feature vector in the sequence, and output the prediction label (real value) distribution;

3. CTC loss(transcription Layer): use CTC loss to convert a series of label distribution obtained from circulation layer into final label sequence.

3.2.2 Structure of CNN in CRNN



Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 × 2, s:1, p:0
MaxPooling	Window:1 × 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
MaxPooling	Window:1 × 2, s:2
Convolution	#maps:256, k:3 × 3, s:1, p:1
Convolution	#maps:256, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:128, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:64, k:3 × 3, s:1, p:1
Input	$W \times 32$ gray-scale image

Figure 12: Structure of CNN in CRNN

There are four maximum pooling layers in total, but the window size of the last two pooling layers is changed from 2x2 to 1x2, that is, the height of the image is halved four times (divided by 2^4), and the width is halved only two times (divided by 2^2). This is because most text images are smaller in height and longer in width, so their features Map is also a rectangle shape of high, small, wide and long. If 1 Œ 2 pooling window is used, information in the width direction can not be lost as much as possible. It is more suitable for English letter recognition (such as distinguishing I and L).
CRNN also introduces the batchnormalization module to accelerate the model convergence and shorten the training process.
The input image is grayscale image (single channel); the height is 32, which is fixed. After the image passes CNN, the height becomes 1; the width is 160 or other values, but it needs to be unified, so the data size of input CNN is (channel, height, width) = (1, 32, 160).
The output size of CNN is (512, 1, 40). That is to say, CNN finally gets 512 feature maps, each of which has a height of 1 and a width of 40.

3.2.3 Structure of RNN(LSTM) in CRNN

Because RNN has the problem of gradient disappearing and cannot get more context information, the special design of LSTM allows it to capture long-distance dependency.
LSTM is one-way, it uses only past information. However, in image-based sequences, the context of the two directions is useful and complementary to each other. So we combine two lstms, one forward and one backward into one bidirectional LSTM. Here, two layers of BLSTM network (256 unit) are used:

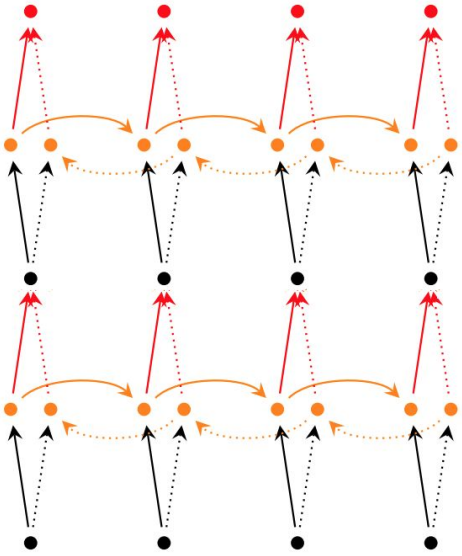


Figure 13: stack bidirectional dynamic rnn

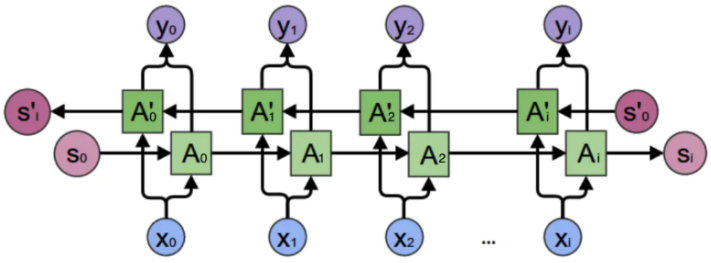


Figure 14: BLSTM in CRNN

Through the above step, we get 40 feature vectors, each of which has a length of 512. In LSTM, one time step passes in one feature vector for classification. There are 40 time steps in total. We know that a feature vector is equivalent to a small rectangular area in the original image. The goal of RNN is to predict which character this rectangular area is. That is, according to the input feature vector, we can predict and get the softmax probability distribution of all characters. This is a vector with the length of the number of character categories as the input of CTC layer.
Because each time step will have an input eigenvector X^T, and output a probability distribution Y^T of all characters, the output is a posterior probability matrix composed of 40 vectors with the length of the number of character categories. As shown in the figure below:

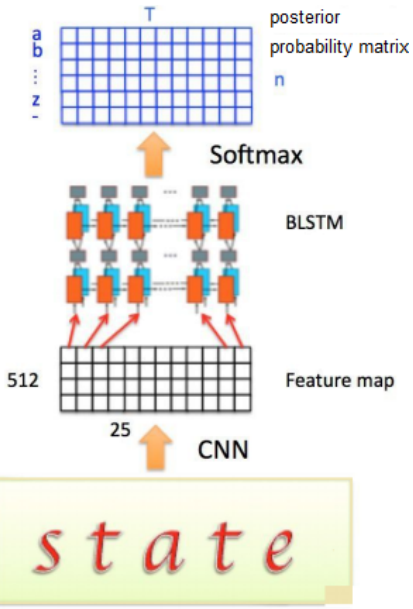


Figure 15: obtain the posterior probability matrix

The posterior probability matrix is then put into the transcription layer.

3.2.4 CTC Loss

Transcription is the process of converting RNN prediction to tag sequence. Mathematically, transcription is to find the tag sequence with the highest probability combination according to the prediction of each frame.
The difficulty of end-to-end OCR recognition lies in how to deal with the alignment problem of indefinite length sequence. OCR can be modeled as temporal dependent text image problem, and then the loss function of CTC (connectionist temporal classification) is used to train CNN and RNN end-to-end.

- **sequence merging mechanism**
Now we need to translate the output sequence of RNN into the final recognition result. When RNN classifies the sequence, there will inevitably be a lot of redundant information, such as a letter is recognized twice in a row, which requires a set of de-redundancy mechanism.

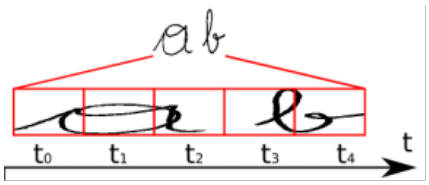


Figure 16: example of sequence merging

For example, we need to recognize the above text. There are five time steps in RNN. Ideally, T0, T1, T2 should be mapped to "a", T3, T4 should be mapped to "b", and then connect these character sequences to get "aaabb". Then we combine the consecutive and repeated characters into one, and the final result is "ab".
This seems to be a better way, but there is a problem. If it's 'book', 'hello' and other words, you will get 'bok' and 'helo' after merging consecutive characters. So CTC has a blank mechanism to solve this problem.
We use the "-" symbol to represent blank, when RNN output sequence, insert a "-" between the repeated characters in the text label. For example, if the output sequence is "bbooo-ook", then it will be mapped to "book", that is, if there is a blank character separation, the consecutive identical characters will not be merged.
That is to say, to delete the continuous repeated characters first, and then delete all "-" characters from the path. This is called the decoding process, while encoding is realized by neural network. By introducing the blank mechanism, we can solve the problem of repeated characters.

- **training step**
In the training step, we need to get the loss function according to these probability distribution vectors and the corresponding text labels to train the neural network model.

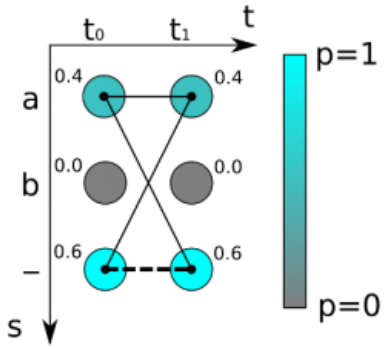


Figure 17: CTC translation/ where the thin line presents path of "a" and the thick dotted line presents path of empty text

As shown in the figure above, for the character recognition with the simplest time sequence of 2, there are two time steps (T0, T1) and three possible characters of "a", "c" and "-". We get two probability distribution vectors. If the maximum probability path decoding method is adopted, the probability of "-" is the largest, that is, the probability of the real character being empty is $0.6 * 0.6 = 0.36$.

But in the case of character "a", there are many alignment combinations, "aa", "a -" and "- a" all represent "a", so the probability of output "a" should be the sum of three kinds:

$$0.4 * 0.4 + 0.4 * 0.6 + 0.6 * 0.4 = 0.16 + 0.24 + 0.24 = 0.64$$

So the probability of "a" is higher than that of "empty". If the label text is "a", the loss function is calculated by calculating the sum of the scores of all possible alignment combinations (or paths)

that are "a" in the image.

So for RNN, the given input probability distribution matrix is $y = Y1, Y2, ..., YT$, T is the sequence length, and the total probability mapped to label text L is:

$$p(l|y) = \sum_{\pi:B(\pi)=1} p(\pi|y)$$

Where $B(\pi)$ represents all path sets of text l after the transformation of mapping function B from sequence to sequence, and π is one of them. The probability of each path is the product of scores of corresponding characters in each time step.
We just need to train the network to maximize the probability value, similar to the common classification, the loss function of CTC is defined as the negative maximum likelihood function of probability. For the convenience of calculation, the likelihood function is logarithmic.
Through the calculation of the loss function, the previous neural network can be back-propagation. The parameters of the neural network are updated according to the optimizer used, so as to find the most possible character corresponding to the pixel area. By mapping transformation and the sum of all possible path probabilities, CTC does not need to segment the original input character sequence accurately.

- **test step**
We use the trained neural network to recognize the new text image. At this time, we do not know any text in advance. If we calculate all the paths of every possible text as above, for a long time step and a long character sequence, the calculation is very large, which is not a feasible solution.
We know that the output of RNN in each time step is the probability distribution of all character categories, that is, a vector containing the score of each character. We take the character with the maximum probability as the output character of this time step, and then splice all time steps to get a character to get a sequence path, that is, the maximum probability path. Then according to the merging sequence method described above, we can get the final prediction text results.
In the output stage, after CTC translation, the sequence feature information learned from the network is transformed into the final recognition text, and then the whole text image can be recognized.

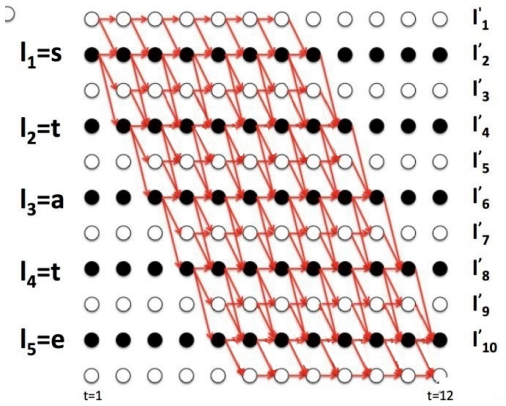


Figure 18: final recognition process

For example, in the above figure, there are five time steps, and the character categories are "a", "b" and "-" (blank). For the probability distribution of each time step, we take the character with the largest score, so we get the sequence path "aaa-b", first remove the adjacent repeated characters to get "a-b", then remove the blank character to get the final result: "ab".

3.2.5 Summary of CRNN

In the process of prediction, we first use standard CNN network to extract the features of the text image, then use BLSTM to fuse the feature vectors to extract the context features of the character

sequence, and then get the probability distribution of each column of features. Finally, we predict the text sequence through the transcription layer (CTC).
Blstm and CTC are used to learn the context in the text image, so as to effectively improve the accuracy of text recognition and make the model more robust.
In the training phase, CRNN uniformly scales the training image to 160 Œ 32 (w Œ h); in the test phase, for the problem that character stretching will reduce the recognition rate, CRNN keeps the size proportion of the input image, but the image height must be unified to 32 pixels, and the size of convolution feature map dynamically determines the sequence length (time step) of LSTM.

3.3 Evaluation Protocol

The f-measure is used as the metric. A correct (true positive) result if it has a correct detection of a text box with correct transcription. A detection is counted as correct if the detected bounding box has more than 50% overlap (intersection over union, IOU) with the ground truth box. For example, suppose a ground truth bounding box is $A(g)$ with transcription $t(g)$, and a detected bounding box is A(d) with transcription $t(d)$.
Then a positive match is counted if $(g; d)$ verifies the following condition:

$$\frac{A(g) \cap A(d)}{A(g) \cup A(d)} > 0.5 and t(g) = t(d)$$

where the equation between $t(g)$ and $t(d)$ means the edit distance between the two transcriptions is zero.
At the whole test set level, the evaluation metrics are computed cumulatively from all the test images. Denote the set of positive matches as M, the set of expected words as G and the set of filtered results as T. Then the f-measure is computed as follows:

$$P = \frac{|M|}{|T|}$$

$$R = \frac{|M|}{|G|}$$

$$f - measure = \frac{2 * P * R}{P + R}$$

4 EXPERIMENTS

4.1 Detection–CTPN

We randomly choose $\frac{1}{10}$ of training dataset (794 images) for validation, and other $\frac{9}{10}$ of training images for training models. We record the loss values (including cls loss,rgr loss and the total loss) for each iteration. Because of time limitation, we only train for 22 epoches.

4.1.1 training

In the 22 epoches, we record the loss value every 200 iteration. As we have approximately 7200 images, there are 35 steps in each epoch. It is obvious to view a significant decline on all the loss value.

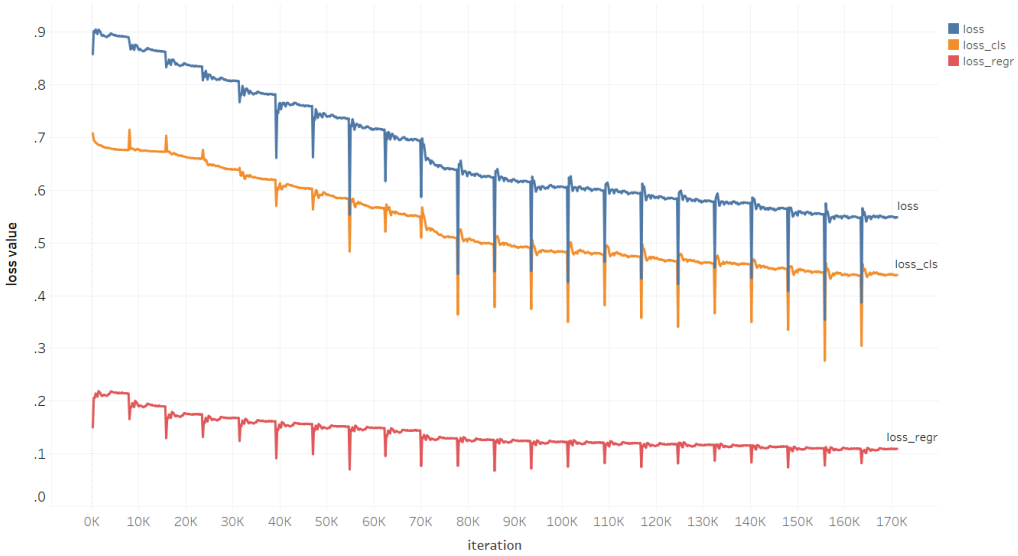


Figure 19: loss value in each iteration

Also, we can make the boxplot of each epoch. We use the intensity of color to present step number(1-35). We can see the first step in each epoch always has the least loss value–it is probably related to the image feature itself in the fist step.

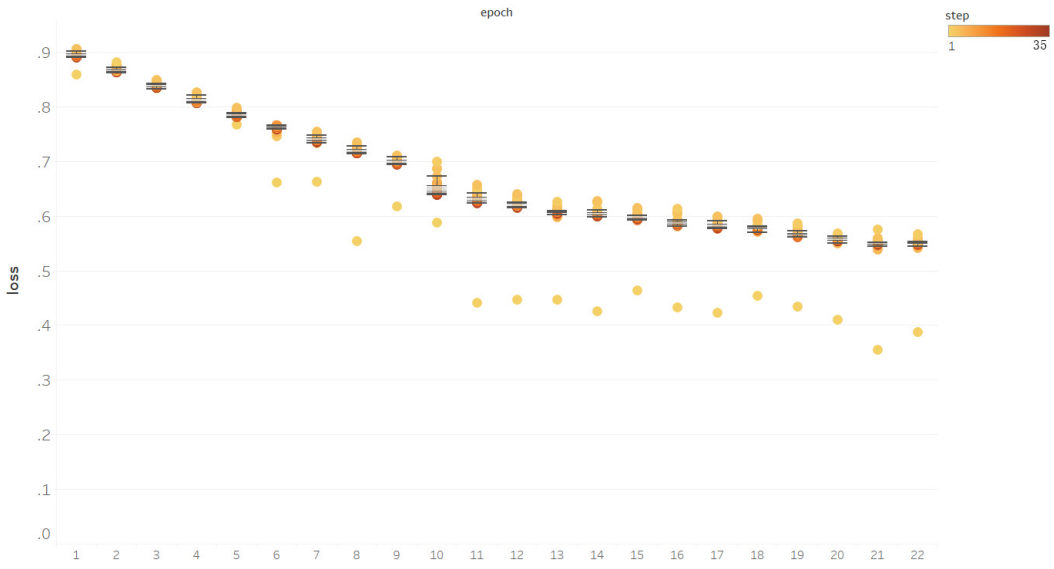


Figure 20: loss value in each epoch

4.1.2 evaluating

After building the model, We detect text proposals from the image and draw the box contour of them.

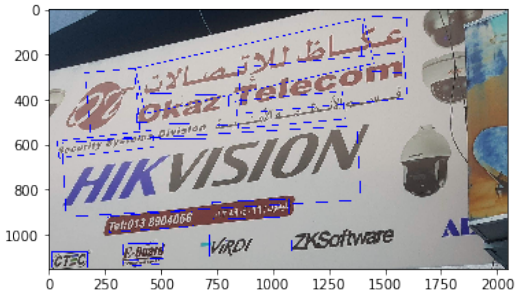


Figure 21: box line of our prediction result

We collected the number of ground truth, the number of our prediction result and the number of valid prediction result (which fits the IOU condition) in all the images.
As we can see, the valid prediction number rises as the epoch increases.

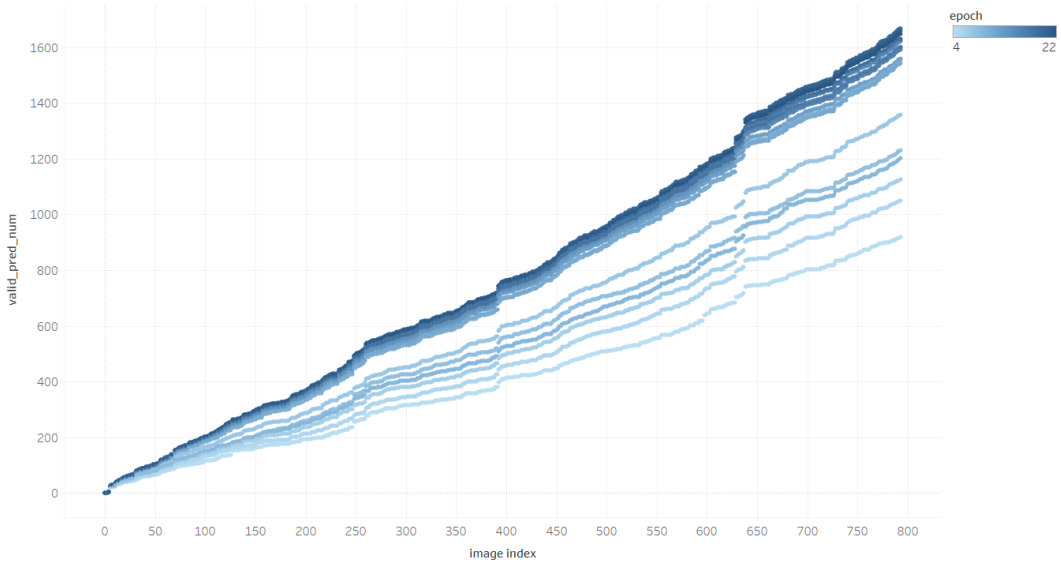


Figure 22: valid prediction number in different epoch

Here we present the numbers compared in the last epoch as an example. There are 8356 ground-truth words in 794 images. And using our model, we can detect and segment 4247 pieces of text(5.35 pieces one image averagely).

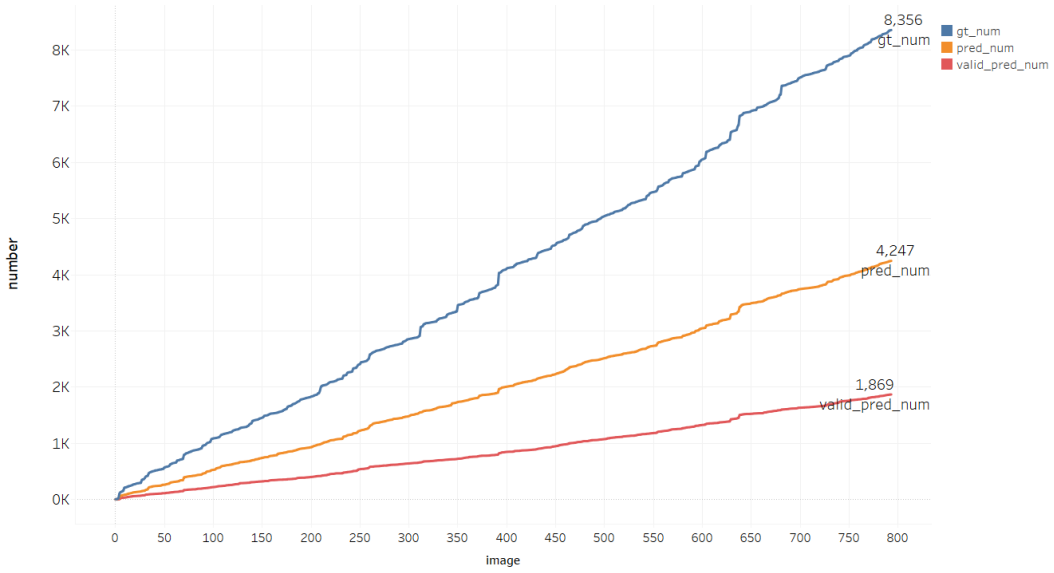


Figure 23: number compared in the last epoch

However, only 1869 of them are counted as correct using the intersection over union(IOU) method. In some cases, our model is not able to segment ground-truth words from the text line. So the IOU may be less than 0.5 and one missing detection will ensue.
For example, we can detect the telephone number (in tr_img_00001.jpg) as the text line in our model, but the ground truth would like to split the text in two parts according to the space.

271,943,434,920,431,973,269,983,Latin,Tel:013
449,923,648,895,636,951,446,968,Latin,8904066

Figure 24: detection in our model

This problem may be solved in the RECOGNITION stage. If we recognize more than one ground truth words in one text line we segmented, we can separate them and change the detection result. We believe the performance could be improved in a large measure if still using this evaluation protocol.

As the Evaluation Protocol stated, we can calculate the f-measure of each epoch.

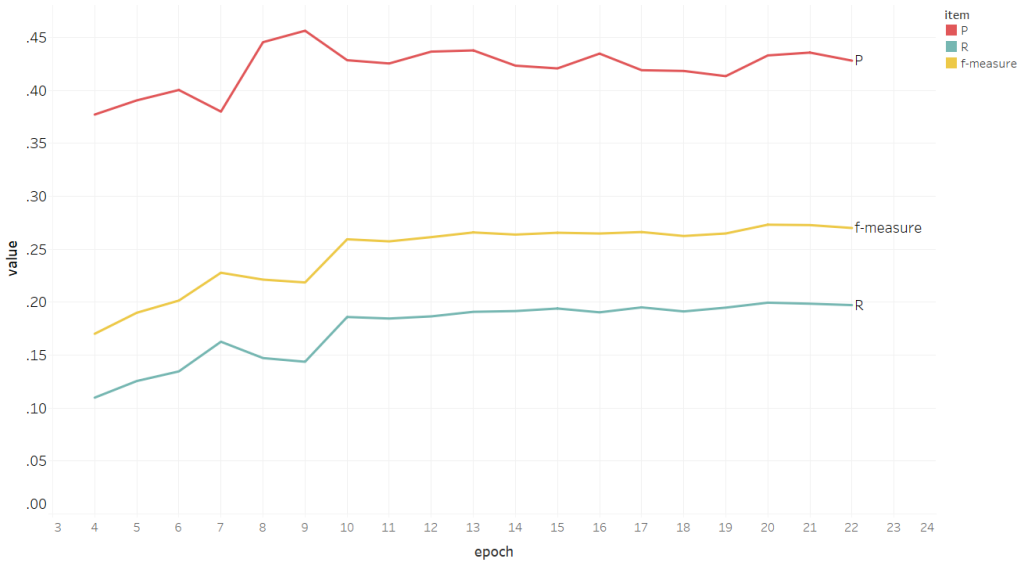


Figure 25: f-measure of detection

In the last epoch:

$$P = \frac{1869}{4247} = 0.440$$

$$R = \frac{1869}{8356} = 0.224$$

$$P = \frac{2 * 0.440 * 0.224}{0.440 + 0.224} = 0.297$$

The final result of detection stage is 0.297.

4.1.3 testing

We segmented text lines of the 1990 test images and obtained 8944 text lines(5.00 pieces one image averagely).

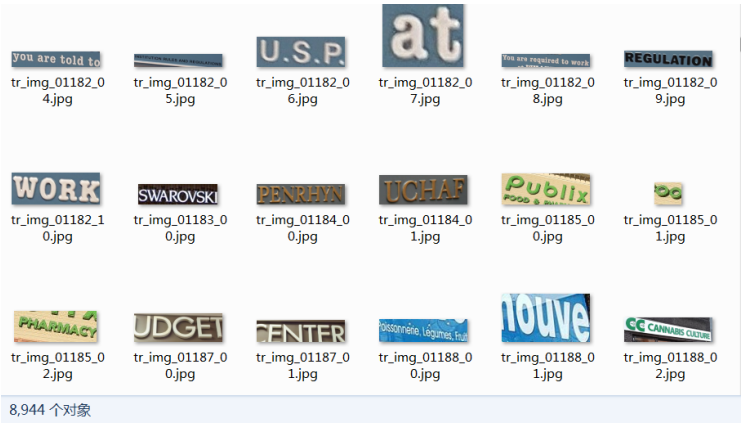


Figure 26: detection in our model

Something worth to be mentioned is that, images in a large size are vulnerable to trigger the cuda error. So we can try to resize them to half of the original size to avoid the error.

4.2 Recognition–CRNN

4.2.1 data processing

The original data set is the whole picture. In order to obtain the training input of CRNN model, we crop all pictures according to the position of text content according to the GT file in advance, get the main scene text picture, and process it as a standard rectangular image. The text picture name and text picture content are stored in the text file.

Among them, text content of some pictures is wrong. For example, the width or height of the text picture is less than one pixel, or the annotation position of the text picture is beyond the scope of the original picture. After data cleaning, only the pictures with compliant position annotation are kept.

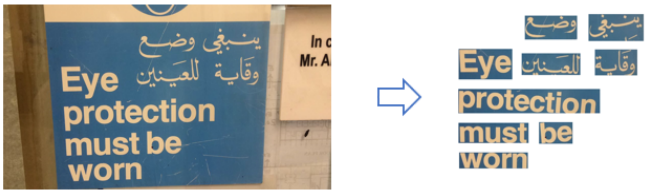


Figure 27: data input in CRNN

4.2.2 training

Considering that the training set and test set contain a large number of Chinese, Korean, Japanese characters etc, we tend to choose a window with a length-width ratio of 1:1. Here we choose a 2*2 window.

Gradient explosion occurs frequently in our training. We come up with many methods to weaken the bad effect. And due to the fact that there is no zero_infinity parameter in Pytorch 1.0.0 to avoid gradient explosion.

At the beginning of model training, there will be gradient explosion at the first or second epoch. Because there are so many layers of CRNN, and gradient explosion is easy to occur due to the accumulation of gradient in the process of back propagation. In order to solve this problem, we try the following measures:

1. Use LSTM after CRNN to suppress the occurrence of gradient explosion

2. Using relu function as activation function

3. Add batchnorm layer after each convolution layer in the model

4. Reduce learning rate to 5e-5

5. Reduce the batch size

6. Convert the size input image into 512 * 32

7. Perform gradient truncation and set the threshold value to 1e8. When the gradient exceeds the threshold value, it will be limited to 1e8

After a series of attempts, the model can be trained normally in more than ten epochs, but there will still be gradient explosion after that, which is possible as follows:

1. The number of alphabets is too large. When the setting weight is not good enough, the probability may be extremely small. Thus it is easy to generate gradient explosion after derivation.

2. Even if gradient truncation is used, gradient explosion may still occur in multi-layer perceptron network and LSTM with long input sequence.

4.2.3 evaluating

Use validation dataset to evaluate the performance. We can see the accuracy is only 1.34%.

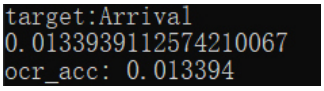


Figure 28: CRNN accuracy

The performance of the model(the accuracy of the text recognition in the valid dataset) is not good enough is largely due to two few epoches we have trained. And we believe that it can perform much better if the gradient explosion problem can be solved.

5 SUMMARY

In this paper, we use CTPN for detection and CRNN for recognition. The detection stage can obtain a f-measure of 0.297. However, due to frequent gradient explosion and time limitation, we cannot get satisfied result in the recognition stage as expected.

There are still many things we can do in the future:
1. we used vertical anchors to detect horizontal text. In the future, we can try to add horizontal anchors in an attempt to detect vertical text.
2. modify network structure and use residual network.
3. adjust the sharpness and contrast of the input picture.
4. try the data augment and expand the training set to improve the generalization ability of the model.
5. try to train model in different layers and process fine-tuning after pre-training.

6 REFERENCE

[1]Zhi Tian, Weilin Huang, Tong He, Pan He,Yu Qiao. Detecting Text in Natural Image with Connectionist Text Proposal Network.Computer Vision-ECCV 2016-14th European Conference, Amsterdam,The Netherlands, October 11-14, 2016

[2]Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE transactions on pattern analysis and machine intelligence, 2016.

[3]Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, 2015.

[4]Yuchen Dai, Zheng Huang, Yuting Gao, Youxuan Xu, Kai Chen, Jie Guo, and Weidong Qiu. Fused text segmentation networks for multi-oriented scene text detection. In 2018 24th International Conference on Pattern Recognition (ICPR), 2018.

[5]Nibal Nayef, Cheng-Lin Liu, *et al*.ICDAR2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition - RRC-MLT-2019. ICDAR 2019: 1582-1587

[6]https://zhuanlan.zhihu.com/p/43534801 ; https://blog.csdn.net/bestrivern/article/details/91050960; https://www.jianshu.com/p/4ac876a4cd5c