Deep Learning Neurl Network Final Report

School of Data Science
RUIPU LUO
16307130247
CHENG ZHONG
16307110259

Abstract

In this final project, we used a pipeline model for OCR tasks, that is, first detect the position of the text box, and then do the recognition of text in the box.For the text detection stage, we compared the YOLOv3 model and the EAST model, which are widely used in object and text recognition. Due to the high resolution of the picture in the training set, we replaced the backbone of the original EAST model with resnet18, and achieved a certain performance improvement.In the text recognition stage, we adopted two models that is CRNN and Seq2Seq with Attention, which are the most widely used. We finally used the model combination of **YOLOv3+CRNN** and obtained the best f-measure:(**0.3055**) on the validation set, which is split from the training set.

1 Introduction

Optical Character Recognition (OCR) refers to the process of analyzing and recognizing image files of text data to obtain text information. That is to recognize the text in the image and return it as text.

According to the recognition environment, According to the recognition scenario, we can divide OCR into special OCR for specific scenarios and general OCR for multiple scenarios.For example, the widely used ID card recognition and car license plate recognition are typical examples of dedicated OCR. General OCR can be used in more complex scenarios and also has greater application potential. However, because the scene of the general picture is not fixed and the text layout is diverse, it is more difficult. According to the content of the recognized pictures, the scenes can be divided into simple scenes with clear and fixed patterns and more complex natural scenes.

Natural scene text recognition is extremely difficult, because the background of the picture is extremely complex, and it often faces the problem of low brightness, low contrast, uneven lighting, perspective distortion and incomplete occlusion. Sometimes the layout of the text may have problems such as distortion, wrinkling, and reversal, and the text in it may also have various fonts and different sizes and colors .

Typical OCR processing steps are shown in Figure1. In these steps, the technical bottlenecks affecting recognition accuracy are text detection and text recognition, and these two parts are also the top priority of OCR technology.

In traditional OCR technology, image preprocessing is usually to correct the image. Common steps of processes include: geometric transformation (perspective, distortion, rotation, etc.), distortion correction, blur removal, image enhancement, and light correction, etc.

Text detection is to detect the location and scope of text and its layout. Usually also includes layout analysis and text line detection. The main problem to be solved by text detection is where there is text and how wide the text is.

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

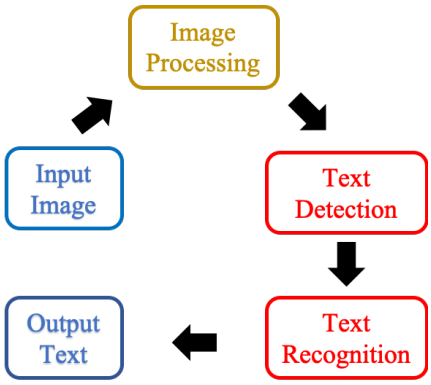

Figure 1: Traditional OCR process

Text recognition is to recognize the content of text on the basis of text detection and convert the text information in the image into pure text. The main problem solved by text recognition is what each text is. The recognized text usually needs to be checked again to ensure its correctness. Text correction is also considered to belong to this link. When the recognized content is composed of the words in the lexicon, we call it Lexicon-based, otherwise it is called Lexicon-free. The task of this project belongs to Lexicon-based.

In the text detection stage, we uses the EAST[12] model and the YOLOv3[6] model for comparison. And in the text recognition stage, we use CRNN with CTCLoss[7] and attention based CRNN with Seq2Seq model[11] for comparision.

2 Related Works

2.1 Text Detection

In the text detection stage, there have been many models that can detect text areas in most scenes.The goal of the text detection model is to find the area of the text as accurately as possible from the picture.

However, the conventional object detection methods in the visual field (SSD, YOLO, Faster-RCNN, etc.) directly applied to the text detection task are not ideal. Figure 2 shows some different of text area detection and target detection. The main reasons are as follows:

(1) Compared with conventional objects, the length of text lines and the ratio of length to width vary greatly.

(2) The text line is directional. The amount of information about the description method of the four-tuple of the regular object frame BBox is insufficient.

(3) The partial images of some objects in natural scenes are similar to the shape of letters, and there will be false alarms if the global information of the images is not referenced.

(4) Some artistic fonts use curved lines of text, and there are many variations of handwriting fonts.

(5) Due to the rich background image interference, hand-designed features are not robust enough in natural scene text recognition tasks.

In response to the above problems, various technical solutions based on deep learning have emerged in recent years. They have modified conventional object detection methods from the perspectives of feature extraction, regional recommendation network (RPN), multi-objective collaborative training, Loss improvement, non-maximum suppression (NMS), and semi-supervised learning, which greatly improves natural scene text detection Accuracy.Here are some text detection models that have been proposed.

CTPN[10] is currently the most widely used and most influential open source text detection model, which can detect horizontal or slightly oblique text lines. The text line can be regarded as a character

Figure 2: some examples of text detection and object detection

sequence, rather than a single independent target in general object detection. Each character image on the same text line can be a context with each other. In the training phase, let the detection model learn this kind of context statistics contained in the image, which can effectively improve the prediction accuracy of the text block in the prediction phase.

RRPN (Rotation Region Proposal Networks)[5] incorporates rotation factors into classical regional candidate networks (such as Faster RCNN). In this proposed model, the ground truth of a text area is expressed as a rotating border with tuples (x, y, h, w,), the coordinates (x, y) represent the geometric center of the border, and the height h is set to The short side of the frame, the width w is the long side, and the direction is the direction of the long side.

The FTSN[2] (Fused Text Segmentation Networks) model uses segmentation networks to support slanted text detection. It uses Resnet-101 as the basic network and uses multi-scale fusion feature maps. Annotated data includes pixel masks and borders of text instances, and uses pixel prediction and border detection for multi-target training.

In DMPNet (Deep Matching Prior Network)[4], the author uses quadrilateral (non-rectangular) to mark the boundaries of text regions more compactly. The model trained by this method has better detection effect on oblique text blocks.

In the EAST (Efficient and Accuracy Scene Text detection pipeline)[12] model, the author first uses a fully convolutional network (FCN) to generate multi-scale fusion feature maps, and then directly performs pixel-level text block prediction on this basis. In this model, two types of text area labeling are supported: rotating rectangular frame and arbitrary quadrilateral. During model prediction, each pixel in the feature map is predicted to its distance to the four sides of the rectangular frame and the direction angle of the rectangular frame.

2.2 Text Recognition

The goal of the text recognition model is to recognize the text content from the segmented text area.

CRNN (Convolutional Recurrent Neural Network)[7] is currently a popular text recognition model that can recognize long text sequences. It contains CNN feature extraction layer and BiLSTM sequence feature extraction layer, which can carry out end-to-end joint training. It uses BiLSTM and CTC components to learn the contextual relationship in character images, 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 BiLSTM 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, and finally make predictions through the transcription layer (CTC rule) Get the text sequence.

The RARE (Robust text recognizer with Automatic Rectification)[8] model works well in recognizing distorted image text. In the model prediction process, the input image must first be sent to a spatial transformation network for processing, and the corrected image is then sent to the sequence recognition network to obtain the text prediction result.

3 Methodology of Text Detection

In this part we will introduce the two text detection models we used for the experiment.

In the text detection stage, we selected two models for comparison. One is YOLOv3, which is widely used in object detection, and the other is EAST, which can detect slanted text. The network structure and principle of the two are very different.

3.1 YOLOv3

As the third generation model of the YOLO series, the YOLOv3 model has greatly improved in training speed and accuracy. Because of hardware limitations, We chose YOLOv3 as one of the models of the first stage, which is also considered from these two aspects above.

3.1.1 Model Structure

Yolov3 uses Darknet-53 as the backbone of the entire network. The entire network structure is not given in the Yolov3 paper.We get the entire YOLOv3 network structure according to the network settings provided in the code as shown in Figure 3.

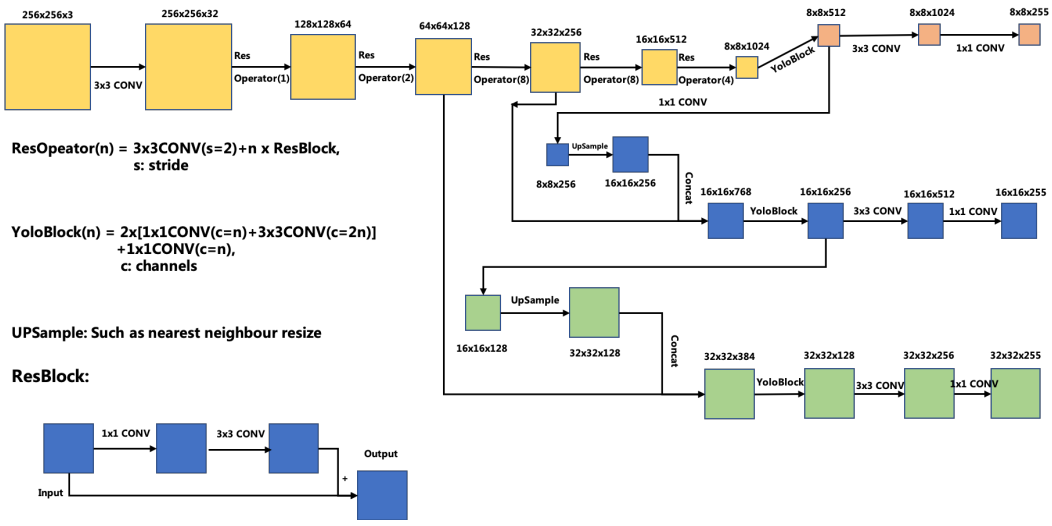

Figure 3: The structure of Yolov3

In Yolov3, there is only convolutional layers, and the size of the output feature map is controlled by adjusting the convolution stride step size. Therefore, there is no particular limitation on the input image size.

Yolov3 draws on the idea of pyramid feature maps. Small size feature maps are used to detect large size objects, while large size feature maps detect small size objects. The output dimension of the feature map is

$$N \times N \times [3 \times (4 + 1 + numberofclass)]$$

$N \times N$ is the number of output feature grid points, and it has 3 Anchor boxes, each box has a 4-dimensional prediction box value t_x, t_y, t_w, t_h, and 1-dimensional prediction box confidence score.

Yolov3 outputs a total of 3 feature maps. The first feature map is downsampled by 32 times, the second feature map is downsampled by 16 times, and the third feature map is downsampled by 8 times. The input image passes through Darknet-53, and then the feature map generated by Yoloblock has three roles. The first role is to generate feature map 1 after some convolution operations. And the second role is to stitch the intermediate output layer of the Darnet-53 network through the 1x1 convolution layer and the upsampling layer to generate feature map 2. The process of generating feature map 3 is similar.

Yolov3's entire network has absorbed the advantages of Resnet, Densenet, and FPN. It can be said to be the fusion of all the most effective techniques for object detection.

3.1.2 Forward Process

According to different input sizes, output feature maps of different sizes will be obtained. Taking an input picture of size $256 \times 256 \times 3$ as an example, the shape of the output feature map is $8 \times 8 \times (3 \times (5 + classnumbers))$, $16 \times 16 \times (3 \times (5 + classnumbers))$, $32 \times 32 \times (3 \times (5 + classnumbers))$. In the design of Yolov3, each grid of each feature map is configured with 3 different prior anchors, that is 3 prior detected boxes.The illustration of feature map is shown in Figure 4.

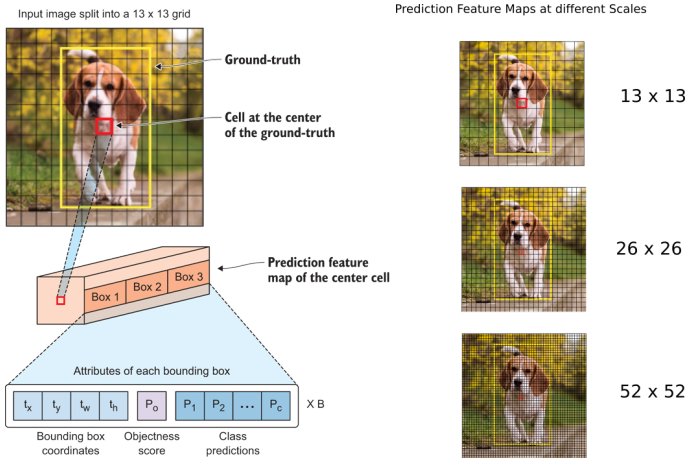

Figure 4: the illustration of feature map

After we have these prior anchor boxes, we can decode these prior boxes to get our final detection boxes, and get the center(x,y), height(h) and width(w) of the detection boxes. The decoding formula is as follows:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_w e^{t_h}$$

As shown in the figure 5, $\sigma(t_x), \sigma(t_y)$ is the offset based on the grid coordinates of the upper left corner of the center point of the rectangular frame, σ is the activation function, and the author uses Sigmoid in the paper. p_w, p_h is the width and height of the prior anchors. Through the above formula, the width and height of the actual prediction frame b_w, b_h are calculated.

The detection confidence score of objects is very important in Yolo design, which is related to the detection accuracy and recall rate of the algorithm. The confidence score is fixed in the output dimension $(4+1+numberofclass)$, and it can be decoded by the Sigmoid function. After decoding, the confidence score's range is in [0, 1].

3.1.3 Backward Process

Yolov3's training strategy is particularly important. The prediction frame is divided into three cases: positive case, negative case, ignore case. The explanation of the positive case is to take any ground

truth in all the labels and calculate the IOU with $N \times N \times 3$ prediction boxes respectively. The prediction box with the largest IOU is the positive case. And a prediction box can only be assigned to one ground truth. When the first ground truth has matched a positive detection box, then the next ground truth will be matched in the remaining $N \times N \times 3 - 1$ prediction boxes.

The ignored cases are these prediction boxes with any ground truth whose IOU is greater than the threshold (0.5 used in the paper), except positive cases. And the ignored cases do not produce any loss.

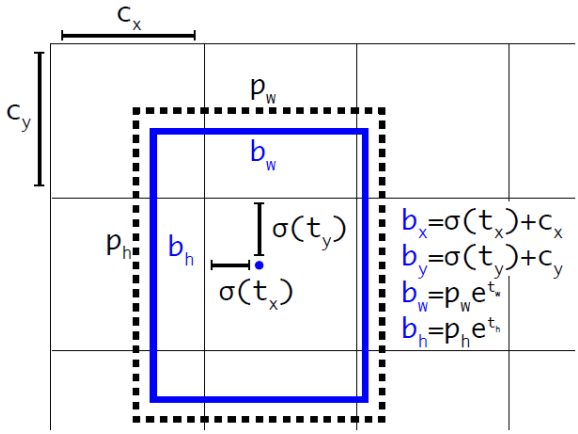

Figure 5:

The negative cases are these prediction boxes with any ground truth whose IOU is less than the threshold (0.5 used in the paper). And the negative cases only produce loss, the confidence score label is 0.

The formula for loss calculation is as follows:

$$loss_{N_1} = \lambda_{box} \sum_{i=0}^{N_1 \times N_1} \sum_{j=0}^{3} 1_{ij}^{obj} \left[(t_x - t'_x)^2 + (t_y - t'_y)^2 \right]$$

$$+\lambda_{box} \sum_{i=0}^{N_1 \times N_1} \sum_{j=0}^{3} 1_{ij}^{obj} \left[(t_w - t'_w)^2 + (t_h - t'_h)^2 \right]$$

$$-\lambda_{obj} \sum_{i=0}^{N \times N} \sum_{j=0}^{3} 1_{ij}^{obj} \log(c_{ij})$$

$$-\lambda_{noobj} \sum_{i=0}^{N_1 \times N_1} \sum_{j=0}^{3} 1_{ij}^{noobj} \log(1 - c_{ij})$$

$$-\lambda_{class} \sum_{i=0}^{N_1 \times N_1}$$

$$\sum_{j=0}^{3} 1_{ij}^{obj} \sum_{c \in classes} \left[p'_{ij}(c) \log(p_{ij}(c)) + \left(1 - p'_{ij}(c)\right) \log(1 - p_{ij}(c)) \right]$$

The final loss is the sum of the loss of the three feature maps:

$$Loss = loss_{N_1} + loss_{N_2} + loss_{N_3}$$

3.2 EAST

Compared with YOLOv3, EAST(Efficient and Accuracy Scene Text) is relatively simple. EAST uses a fast and accurate lightweight scene text detection pipeline, which has only two stages. The first stage of the pipeline is based on the Fully Convolutional Network (FCN) model, which directly generates text box predictions. To produce the final result the second stage is to generate non-maximum suppression of the generated text prediction boxes ((which can be rotated rectangles or horizontal rectangles)). The model gives up unnecessary intermediate steps and performs end-to-end training and optimization in text detection.

3.2.1 Model Structure

Because the size of the text area varies greatly, positioning large text will require deeper features (large receptive fields), while positioning small text requires shallow features (small receptive fields). Therefore, the network must use different levels of features to meet these requirements, but merging a large number of channels on a large feature map will significantly increase the post-calculation cost. To compensate for this, EAST uses the idea of U-shape to gradually merge feature maps while keeping the upsampling branch small. In this way, establishing a network together can not only use different levels of features, but also save little computing cost.The model structure is shown in Figure 6. The network structure can be decomposed into three parts: feature extraction stage, feature

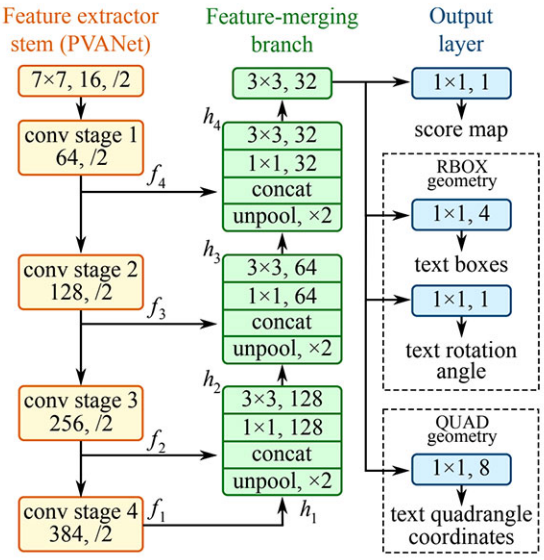

Figure 6: the Network Structure of EAST

merging stage and output stage.

The feature extraction stage is responsible for extracting image features. First, use the pre-trained convolutional network parameters to initialize the network; then, based on the VGG16 model, extract four levels of feature maps (denoted as f_i) from the feature extraction stage, the size of which is $1/32$,$1/16$,$1/8$ and $1/4$ of the input image . In addition, at this stage, the feature maps after $pooling-2$ to $pooling-5$ are extracted for feature merging.

In the feature merging stage, the feature maps obtained in the feature extraction stage are merged layer by layer.In each merge stage, first, the feature map from the previous stage is first input to an $unpooling$layer to expand its size. Then, merge with the current layer feature map.The calculation formula for the merge operation is as follows

$$g_i = \begin{cases} unpool\left(h_i\right) & if \quad i \leq 3 \\ conv_{3\times3}\left(h_i\right) & if \quad i = 4 \end{cases}$$
$$h_i = \begin{cases} f_i & if i = 1 \\ conv_{3\times3}\left(conv_{1\times1}\left(\left[g_{i-1}; f_i\right]\right)\right) & otherwise \end{cases} \tag{1}$$

Finally, the network reduces the number of channels and the amount of calculation through $conv1 \times 1$, and merges the local information through $conv3 \times 3$ to obtain the combined output. After the last merge stage, the $conv3 \times 3$ layer will generate the final feature map of the merged branch and send it to the output stage.

The output layer contains several conv11 convolution operations, which project the feature maps of 32 channels onto a 1-dimensional fractional feature map F_s and a multi-dimensional geometric feature map F_g. The form of the geometric feature map can be either RBOX or QUAD as shown in the table.

Geometry	channels	description
AABB	4	$G = R = d_i \| i \in 1, 2, 3, 4$
RBOX	5	$G = R \| \theta$
QUAD	8	$G = Q = (\Delta x_i, \Delta y_i), i\ in 1, 2, 3, 4$

Table 1: Output Geometry Design

3.2.2 Loss Function

The loss function formula is

$$L = L_{\mathrm{s}} + \lambda_{\mathrm{g}} L_{\mathrm{g}} \tag{2}$$

L_s and L_g represent the loss of the score graph and the geometric graph, respectively, λ_g represents the importance between the two losses.

In order to simplify the training process, this article uses class-balanced cross entropy, the formula is as follows

$$L_{\mathrm{s}} = balanced - xent\left(\hat{\mathbf{Y}}, \mathbf{Y}^*\right) = -\beta \mathbf{Y}^* \log \hat{\mathbf{Y}} - (1-\beta)\left(1-\mathbf{Y}^*\right) \log(1-\hat{\mathbf{Y}}) \tag{3}$$

$\hat{\mathbf{Y}}$= F_s is the prediction of the score graph, and Y^* is the labeled value. The parameter β is the balance factor between positive and negative samples, the formula is as follows

$$\beta = 1 - \frac{\sum_{y^* \in \mathbf{Y}^*} y^*}{|\mathbf{Y}^*|} \tag{4}$$

For geometry loss L_g, it is divided into $RBOX$ Loss and $QUAD$ Loss. $RBOX$ uses IoU loss for the $AABB$ part because it is invariant to objects of different sizes:

$$L_{\mathrm{AABB}} = -\log IoU\left(\hat{\mathbf{R}}, \mathbf{R}^*\right) = -\log \frac{\left|\hat{\mathbf{R}} \cap \mathbf{R}^*\right|}{\left|\hat{\mathbf{R}} \cup \mathbf{R}^*\right|} \tag{5}$$

Next, calculate the loss of rotation angle:

$$L_\theta\left(\hat{\theta}, \theta^*\right) = 1 - \cos\left(\hat{\theta} - \theta^*\right) \tag{6}$$

$\hat{\theta}$ is the prediction of the rotation angle, and θ^* represents the label value. Finally, the overall geometric loss is a weighted sum of $AABBB$ loss and angular loss, the formula is as follows:

$$L_{\mathrm{g}} = L_{\mathrm{AABB}} + \lambda_\theta L_\theta \tag{7}$$

For QUAD loss, the predicted offset of the text rectangle is:

$$\mathrm{C_Q} = \{x_1, y_1, x_2, y_2, \ldots, x_4, y_4\} \tag{8}$$

This part of the loss can be written as

$$L_{\mathrm{g}}=L_{\mathrm{QUAD}}\left(\hat{\mathbf{Q}},\mathbf{Q}^{*}\right)=\min_{\hat{\mathbf{Q}}\in P_{\mathbf{Q}^{*}}}\sum_{\substack{c_i\in C_{\mathbf{Q}},\\ \tilde{c}_i\in C_{\hat{\mathbf{Q}}}}}\frac{smoothed_{L1}\left(c_i-\tilde{c}_i\right)}{8\times N_{\mathrm{Q}^{*}}} \tag{9}$$

Where the normalized term N_{Q^*} is the length of the short side of the quadrilateral, given by:

$$N_{\mathbf{Q}^{*}}=\min_{i=1}^{4}D\left(p_i,p_{(i \bmod 4)+1}\right) \tag{10}$$

3.2.3 Locality-Aware NMS

For the output of EAST, the time complexity of the simple NMS algorithm is $O(n^2)$, where n is the number of candidate boxes, and this time complexity is too high. Therefore, EAST proposes to merge the geometric figures line by line, assuming that the geometric figures from nearby pixels tend to be highly correlated. When merging the geometric figures in the same line, iteratively merge the geometric figure currently encountered with the last merged figure, the improved time The complexity is $O(n)$. After Locality-Aware NMS, we will get the final output.

4 Methodology of Text Recognition

In this part, we will introduce the model we used for the text recognition stage. There are two mainstream technologies for text recognition stage based on deep learning: CRNN OCR and attention OCR. The main difference between two method is how to convert the sequence feature learned by network into the final result. They both use the CNN+RNN network for feature extraction, however, CRNN use CTCLoss for alignment while attention OCR method using attention mechanism.

4.1 CRNN

CRNN contains three structures: CNN layer for extracting the convolutional feature maps; Recurrent layer for extracting the sequence features based on feature maps from CNN; Transcription layer for converting the sequence features into the final recognition result through duplicate removal and alignment. The model structure is shown in Figure.7.

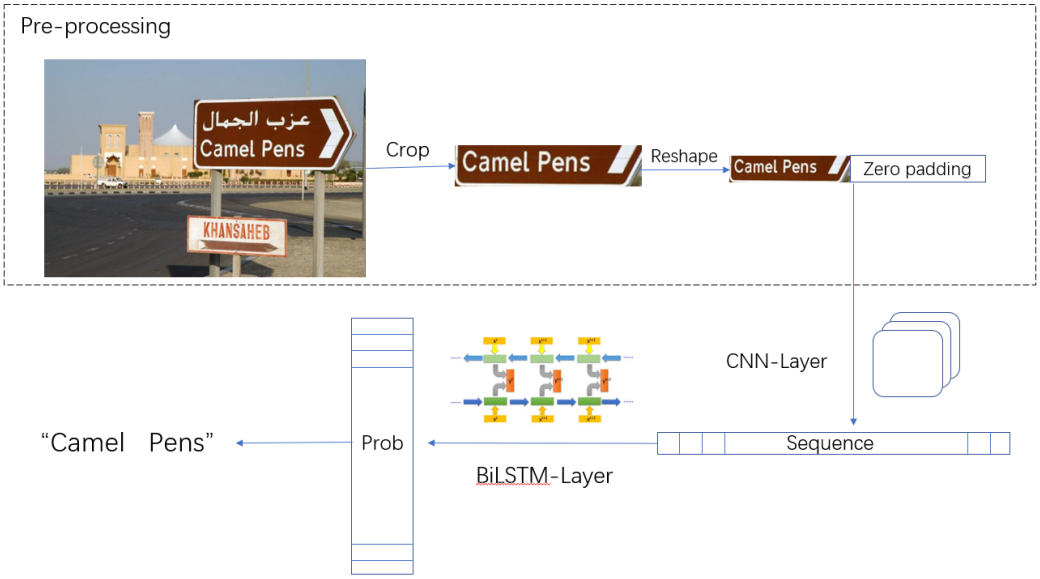

Figure 7: CRNN network

4.1.1 CNN&RNN network

There is no obvious difference between the CNN layer in the CRNN network structure and in other networks. But it should be noted that the inputs are often different in length and width, the pooling layer needs to be adjusted in order to form a serialized output for RNN processing. In general, we set the width of image to 32, and padding the length to a fixed value.

After we convert the image information into sequence information, The image recognition model can be degenerate into a sequence recognition problem similar to speech recognition. For such series data, the LSTM is the most common choice. It can model the context information, and its derivative version BiLSTM can also take the bidirectional semantic dependence. As for image information sequence, it is obvious that the context has a role in recognition, so we use BiLSTM to extract sequence information.

4.1.2 CTC Loss

In sequence learning task, the RNN model generally has a mapping relationship between the input and output sequence in advance. Since the input sequence and the output sequence correspond to each other, the training and prediction of the RNN model can be end-to-end, that is, the loss function of the RNN model can be directly defined according to the difference between the output sequence and the labeled samples.

However, In our training set, it is difficult to divide characters from image, and pictures of the same size may contain different lengths of character sequences. When RNN performs time series classification, a lot of redundant information will inevitably appear. For example, a letter is recognized twice in succession, which requires a method of redundancy mechanisms, but simply remove two same consecutive letters can also cause problems, such as cook and geek, so Alex Graves et.al proposed a end-to-end RNN training method named Connectionist Temporal Classification (CTC)[3]. It allows RNN to directly learn the sequence data without the need to mark the mapping relationship between the input sequence and the output sequence in the training data in advance.

The CTC method using a "blank" mechanism to solve the redundancy problem, the two same consecutive letters will be Merge into one unless it has a "<blank>" inside. e.g. "deep" will be mapped into "dep", while "de<blank>ep" will be mapped into "deep".

In this way, there are many kinds of sequences that can map into the same result. For each sequence π of length T from the input x of BiLSTM, we can find the probability of such sequence generation.

$$P(\pi|x) = \prod_{i=1}^{T} p(x_i) \tag{11}$$

Then for a fixed sequence, its generation probability is the sum of all probabilities of sequence that can be mapped to that sequence, that is:

$$P(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x) \tag{12}$$

where $B^{-1}(l)$ means all sequence that can be mapped to that fixed sequence, so in order to maximize the probability of the correct sequence. we can updated the gradient of BiLSTM through back propagation.

4.2 Seq2seq attention method

For the matching of indefinite length sequences, another idea is to use the Encoder-Decoder structure for matching. The Encoder-Decoder network was proposed by Cho et al [1] and Sutskever[9] in 2014, in order to target the input-output pair of variable-length sequences. Using the encoder to "encode" the input sequence into an abstract message, and then gradually decode the abstract message through the decoder. The so-called encoding and decoding is actually a kind of function mapping, similar to finding a common bridge between two sequence pairs, so that the two sequence pairs correspond.

4.2.1 Seq2seq method

In OCR, the CNN and RNN network are considered as the encoder method, after this case, the image will be encode as sequence for downstream procession. Noted that different from CRNN, the sequence need to be marked to record the beginning and end of sentences.

For text data recognition, we often use LSTM type network as Decoder. In our method, we choose the GRU network with attention method as our decoder since it performs similarly to LSTM but is computationally cheaper. Compared to LSTM, we can use the same gate to forget and choose memory at the same time.

Each step of decoding depends on three variables, one is the source language context vector c obtained from the encoder, which is used to describe the information of the source language, the second is the decoded hidden state vector at the previous moment, and the last is The target word vector predicted at the previous moment is used to show the context dependency of the target language. When the output character is the marker of the end of sentence, the decoding process ends and a new sequence is generated.

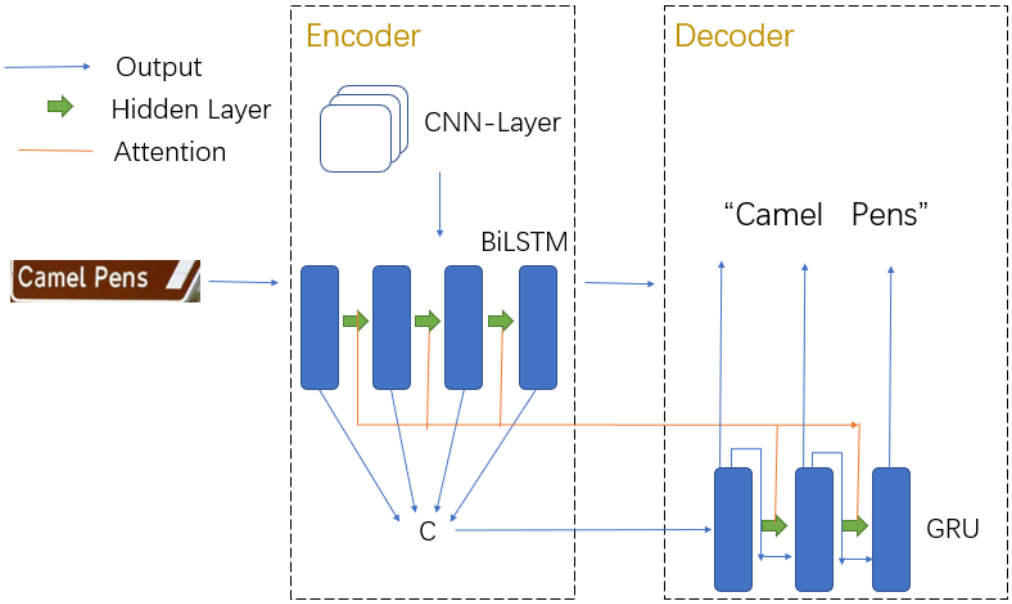

Figure 8: Seq2seq network

4.2.2 Attention method

The above Encoder-Decoder network has a wide range of applications in sequence-to-sequence models. However, there is a problem with the network in the vector c generated by the encoder. Generally speaking, the vector has two limitations. The first is that when the input sequence is very long, the output vector is difficult to express the information of the whole sentence, which is biased to express the information near the end of the sequence; on the other hand, in the process of machine translation, it is necessary to clarify the approximate correspondence between the target language vocabulary and the source language vocabulary. If all the decoding uses the same context vector, it is difficult to show the specific contribution of the input language vocabulary, so now a popular method-Attention mechanism is introduced to solve the problem.

In attention method, we use the state of all hidden layers of Encoder to solve the problem of Context length limitation. When the Decoder outputs characters, its hidden vector and the hidden vector in the Encoder layer are synthesized as a new hidden vector to participate in the output operation. Essentially, it is to weight the decoder's hidden vector according to the encoder's hidden vector.

In this task, beacause we need to extract characters from probabilities (including identifiers at the beginning and end of sentences), we use NLLLoss as loss function. This may lead to poor judgment of the whole sentence.

5 Dataset and Experiment

In this part we will introduce our data set and experimental results.

5.1 Dataset

The data set of this final project is divided into a training set and a test set. There are 7932 images in the training set and 1990 images in the test set. The test set has no ground truth, so we divided 1000 images from the training set for verification. After selecting the optimal model, we trained our final model with the full training set.The training set contains a total of ten languages, they are Arabic, Bangla, Chinese, Devanagari, English, French, German, Italian, Japanese and Korean.

Each ground truth of training data is marked by a 4-corner bounding box and is text transcription. Each picture has a txt file with a corresponding name to store the border and transcript, where each line corresponds to a text block in the image. The format is:

$$x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, script, transcription$$

We also used rotation, flipping, and cropping on the training set to do the data augmentation.

5.2 Experiment

5.2.1 Experiment Setting

The training hyper parameters of YOLOv3 and EAST are shown in table2 and table 3:

parameters	value	parameters	value
epochs	200	batch size	11
number workers	32	image size	416
lr	1e-3	momentum	0.9
lr decay rate	0.1	lr decay step	100

Table 2: hyper parameters of YOLOv3

parameters	value	parameters	value
epochs	300	batch size	32
number workers	32	image size	512
lr	1e-3	momentum	0.9

Table 3: hyper parameters of EAST

The hyper parameters of the two models for text detection are the same, shown in table 4:

parameters	value	parameters	value
epochs	200	batch size	256
number workers	32	LSTM hidden size	256
lr	1e-3	beta1	0.5
lr decay rate	0.5	lr decay step	50

Table 4: hyper parameters of Text Recognition

The training environment of all the above models is $pytorch$ 1.5.0. The graphics card used for training is a $GTX1080ti$.

5.2.2 Text Detection

Because the resolution of many pictures in the dataset is very high, we did not use $VGG-16$ during the training EAST, but adopted $Resnet-21$, so that we will learn more picture features.The training loss landscape of the two models is shown in Figure 8.It can be found that YOLOv3 converges much faster than EAST, which is consistent with the design concept of YOLOv3.

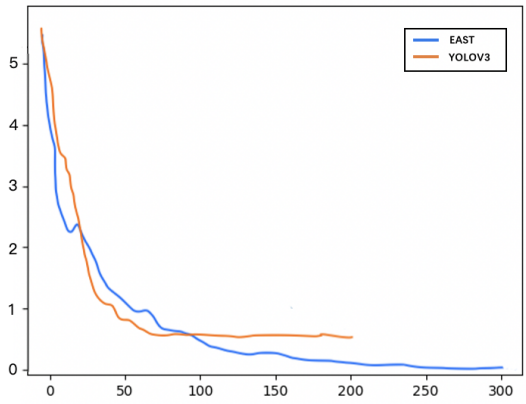

Figure 9: loss landscape for Text Detection

We conducted a comparative experiment on the two models on the validation set. The evaluation metrics of the experiment is f-measure, A detection is counted as correct if the detected bounding box has more than 50% overlap (intersection over union) with the ground truth box.The calculation formula of f-measure is as follows:

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

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

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

Denote the set of positive matches as M, the set of expected words as G and the set of filtered results as T.

The experimental results on the validation set are shown in the table 5.From the experimental results, it can be seen that in the text detection task, the general target recognition model does not perform well in the model that specifically detects text. In terms of f-measure, EAST is 4% higher than YOLOv3.And EAST can detect slanted text, but YOLOv3 can only find the smallest bounding rectangle of slanted text.

Model	Precision	Recall	F-measure
YOLOv3	0.6208	0.4852	0.5447
EAST	**0.6643**	**0.5201**	**0.5834**

Table 5: experimental results in Text Detection

The text box detected by EAST will be slightly smaller than the text, while the text framed by YOLOv3 will be slightly larger than the text itself. Some comparation of detection are shown in Figure 8.

Figure 10: comparison of EAST and YOLOv3. The second line is the test result of YOLOv3

5.2.3 Text Recognition

Because in this project, we are not allowed to use data outside the training set to assist in training, so we can only build a dictionary based on the corpus in the existing training set without other corpus sets. In this way, characters that do not appear in the training set can never be successfully detected in the test set, which increasing the randomness of the test data.
First, we test the model of CRNN, on the validation set, we reached the highest accuracy of **0.4620**. However, after reviewing the relevant data, we found that CTCLoss did not solve the problem of repeated characters very well, and repeated characters still appeared in the wrong recognition results. This may be due to the insufficient of training set, resulting in imperfect learning result.
In this case, we have to leave only one adjacent repeated character in the recognition result. After doing this, the accuracy has increased significantly and reach **0.6918**.
However, the result of the seq2seq + attention model is not ideal. Although its character recognition accuracy rate reaches about 0.3, there are always one or two erroneously recognized words in each word, resulting in a low overall success rate.
Somehow, it also shows that in the case of insufficient training sets, both CTCLoss and NLLLoss do not perform well in the identification of variable-length sequences, especially in de-duplication, but the accuracy of CTCLoss is still better than that of NLLLoss.

Model	CRNN(de-duplication manually)	CRNN(de-duplication automatelly)	Seq2seq+attention
Accuracy	**0.6918**	0.4620	0.3(character)

Table 6: experimental results in Text Recognition

5.2.4 Merge two stage

After merging the two phases, we evaluated the entire pipeline. After merging the two phases, we evaluated the entire pipeline. That is, the text box detected in the first stage is used for recognition with a text recognition model. Since the Seq2Seq with attention model is not very effective, our model only uses CRNN for the text recognition stage.

The evaluation metrics are f-measure, which are similar to the text detection metrics, but the positive examples also need to add the conditions for correct text recognition.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:

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

The final result is as follows: The final result is very unexpected. The prediction result of EAST that

Model	Precision	Recall	F-measure
YOLOv3+CRNN	**0.3034**	**0.3075**	**0.3055**
EAST+CRNN	0.2594	0.2347	0.2464

Table 7: experimental results in Text Detection

performs better in first stage is not as good as YOLOv3 after the text recognition.

We thought about the possible reasons. After comparing the detections of many EAST and YOLOv3, we found that the IOU of many EAST detection and gt is greater than 0.5, but did not completely include the characters. This will result in the inability to decode the text in the text detection stage, but it is correct when calculating the indicator in the first stage.

6 Conclusion and future work

In this report, we compare the performance of two different models of text detection and text recognition on this data set, and finally find that the combination of YOLOv3+CRNN performs best. In future work, we can try to change the loss of YOLOv3 so that YOLOv3 can also detect slanted text. At the same time, we can also explore more effective feature acquisition methods for high-resolution images.

References

[1] Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*, 2014.

[2] 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)*, pages 3604–3609. IEEE, 2018.

[3] Alex Graves, Santiago Fernández, and Faustino Gomez. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine Learning*, 2006.

[4] Yuliang Liu and Lianwen Jin. Deep matching prior network: Toward tighter multi-oriented text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1962–1969, 2017.

[5] Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*, 20(11):3111–3122, 2018.

[6] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[7] 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*, 39(11):2298–2304, 2016.

[8] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4168–4176, 2016.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 2014.

[10] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. In *European conference on computer vision*, pages 56–72. Springer, 2016.

[11] Zbigniew Wojna, Alexander N Gorban, Dar-Shyang Lee, Kevin Murphy, Qian Yu, Yeqing Li, and Julian Ibarz. Attention-based extraction of structured information from street view imagery. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 844–850. IEEE, 2017.

[12] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017.