香港中文大學

計算機科學及工程學系

Department of Computer Science and Engineering,
The Chinese University of Hong Kong

FYP Final Report

Receipt recognition

Name(s)
Choi Chun Kit - 1155094366
Wong Ho Chung - 1155086665

Supervised By

Prof. CHAN Siu On

1 Introduction

1.1 Motivation

With the progress of technology, people start to manage finances on their phones. In recent years, there are many expense tracking apps in the market, which mostly allow users to track their spending automatically by recording users' electronic payments. But for cash payment, users are required to input the information on the receipt to the tracking app manually.

During the day to day usage, we discovered that the procedure of inputting the receipt is complicated. For each receipt, users need to input multiple data including the category of spending, company name, date and the amount. If there are multiple receipts, users need to record those data multiple times, and this costs a lot of effort. We believe the user experience of the current expense tracking app is not friendly enough. We would like to tackle this difficulty by using modern technologies.

Character recognition technique has been developed for a long time. Some significant products, like the Google lens, have mature and powerful text recognition by using deep learning models. With the recent progress on text recognition and artificial intelligence, we believe we can tackle this challenge with deep learning.

1.2 Objective

The goal of this project is to develop an end to end receipt recognition system with multiple deep l earning models. Users can take a photo of the receipt with the mobile camera, then the model will recognize and output the key information including company name, date, address, etc. Our project will focus on the following objectives:

- Develop a text detection model that detect and localize text instances in a picture.
- Develop a text recognition model that recognizes and output text, word, or sentence by inp utting an image of the text instance.
- Develop a an information extraction model that given a text instance (or an image of the te xt instance), the model will determine the class of the text including company, address, dat a, phone number, etc.

1.3 Project Resource

1.3.1 Software and Dataset

We would mainly use Tensorflow 2.0 in the project as it contains lots of resources including articl es, source codes and detailed guides of using Tensorflow 2.0 API [16]. In the training phase, we will train the text detection model based on COCO-text dataset and SynthText dataset which conta ins over 850 thousand images, and train the text recognition model based on MJSynth dataset whic h consists of 9 million images covering 90 thousand English words.

COCO-text dataset

COCO-text dataset is a large-scale dataset for text detection and recognition in natural images. Bas ed on the MSCOCO image data, COCO-text dataset contains over 173 thousand text annotations i n over 63 thousand images for text detection and recognition training [1].

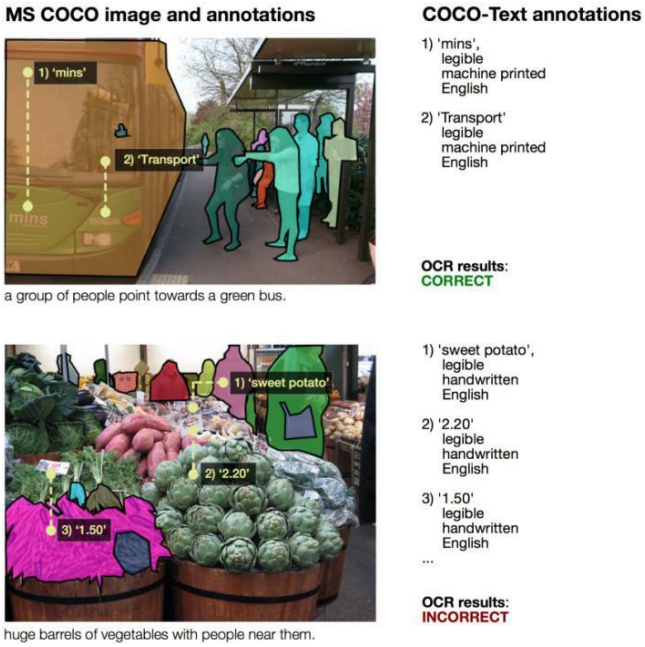

Figure 1. Example of annotations of COCO-text [1]

SynthText in the Wild dataset

SynthText dataset is a synthetically generated dataset, in which word instances are placed in natura l scene images. It contains over 800 thousand images with 8 million word instances [2].

MJSynth dataset

MJSynth (MJ) is a synthetic dataset designed for scene text recognition, containing around 9 milli on word box images. The MJ word box generation process is as follows (see Figure 4): font render ing, border and shadow rendering, or background coloring, composition of font, border, and backg round, applying projective distortions, blending with real-world images, and finally adding noise [3].

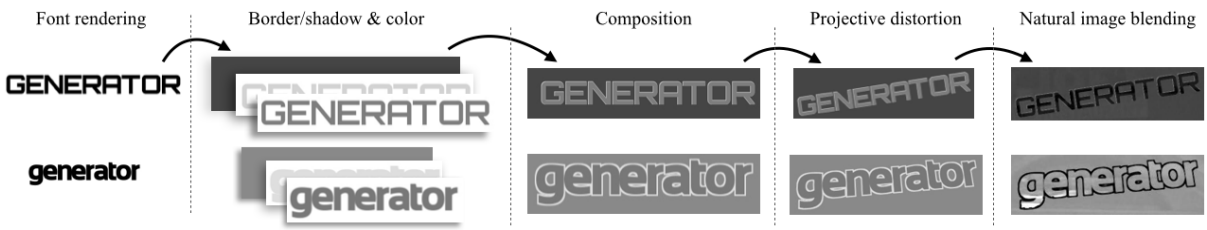

Figure 2. MJSynth dataset: an example "generator" of word box generation process [3]

1.3.2 Literature Survey

In this section, we will discuss the existing researches and methods for text detection and recogniti on.

Rosetta

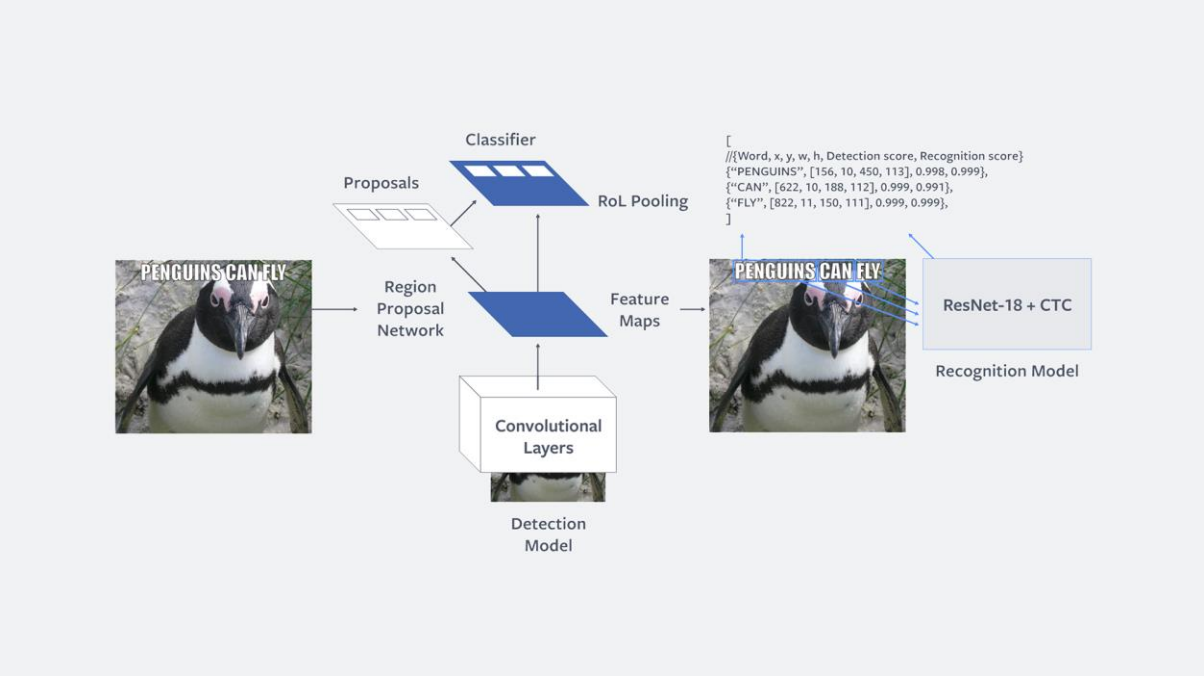

Figure 3. Two-step model architecture of Rosetta [4]

Rosetta model extracts text from billion images from Facebook and Instagram, and input images in to a model that has been trained on Rosetta's classifiers to understand the context of the text and th e image together [4]. Rosetta performs text extraction on an image in two independent steps: Text detection and Text recognition. Base on our device and resources, we would design a similar mode l with Rosetta architecture, containing steps of detection and recognition.

Rosetta: Text detection model

For the text detection model, Rosetta uses an approach based on Faster R-CNN, a state-of-the-art o bject detection model. R-CNN and Fast R-CNN use a selective search method to predefine multipl

e region proposals, region that likely contains an object, using traditional computer vision algorith ms. Then it determines the probability that the region contains an object and the class of the object [10, 11]. Selective search and region proposals generation make R-CNN and Fast R-CNN computa tionally expensive and slow in the training phase.

Instead of generating predefined region proposals outside the model, Faster R-CNN integrated the procedure inside the model with a Region Proposal Network(RPN). First, the image will be passed into convolutional layers and created a set of feature maps, then the region proposal network will p redict a set of region proposals and the objectness scores using such feature maps. The predicted re gion proposals are reshaped using a Region of Interest (RoI) pooling layer which classifies the ima ge within the proposed region and predicts the offset values for the bounding boxes [4, 9].

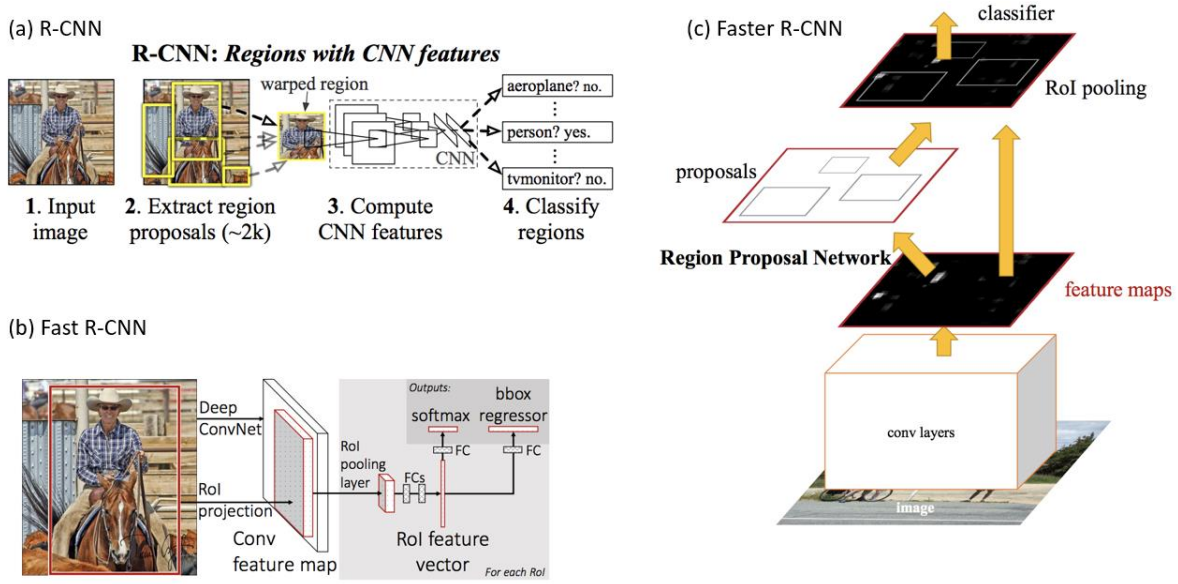

Figure 4. Comparison of the architecture of (a) R-CNN model, (b) Fast R-CNN model (c) Faster R -CNN model for object detection [9, 10, 11]

Rosetta: Text recognition model

For the text recognition model, it is a CNN based on the ResNet18 architecture, as this architecture could get good accuracies and high computation efficiency. Preserving the spatial location of the characters in the image is an important task for word recognition. Therefore, Rosetta modifies Res Net18 architecture by replacing the global average pooling layer with the fully connected layer wit h a convolutional layer at the end of the model, and reducing the strides of the last convolutional l ayers to better preserve the spatial resolution of the features. Furthermore, Rosetta uses long short-term memory (LSTM) units to further improve accuracy. For the loss function, Rosetta would use the connectionist temporal classification (CTC) loss to train the sequence model [4].

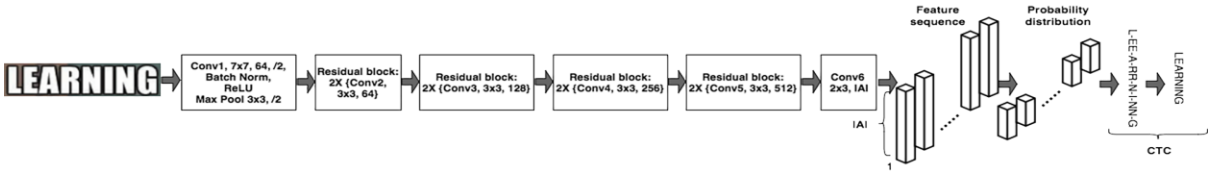

Figure 5. Rosetta architecture of the text recognition model [4]

YOLO (You Only Look Once)

YOLO is another object detection model [13, 15]. Faster R-CNN is a two-stage object detection m odel. It first generates a set of region proposals, then classifies each proposal into different classes [9]. While YOLO is a one-stage object detection model, it can localize and classify the object at th e same time.

From a general perspective, YOLO will divide the image into M x N cells. The cells are responsibl e for predicting the bounding box, the objectness score, and the class scores at the same time. Each cell contains bounding boxes that are centering on that cell. A cell is responsible for predicting the object if the center of the ground truth box is lying on that cell [13, 15].

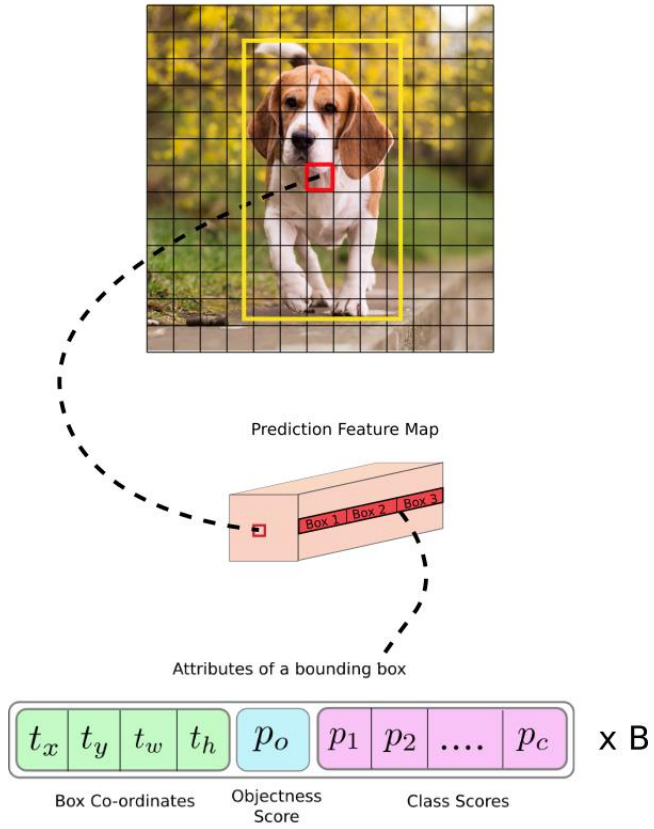

Figure 6. An example of the bounding box in YOLO [15]

In the above picture, the center of the bounding box is inside the red cell. In this case, the red cell i s responsible for predicting that bounding box.

YOLO uses the anchor box prediction method instead of predicting the actual size of the bounding box directly. Anchor box is a predefined bounding box with fixed size and shape. For each box ins ide each cell, the model predicts 4 numbers: t_x, t_y, t_w, t_h, which are the parameters of the anchor box. The box coordinates will be calculated by the following equations:

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

Figure 7. Equations for calculating the bounding box [15]

b_x, b_y are the center coordinate of the bounding box and b_w, b_h are the width and height of the box. c_x, c_y are the offset coordinate from the top-left coordinate (0, 0). p_w, p_h are the width and height of the predefined anchor box. σ is a sigmoid function that normalizes t_x and t_y within the range of 0 and 1 [13, 15].

Each anchor box is associated with an objectness score p_o and class scores $p_c = \{p_1, p_2, \dots\}$. The overall score of an anchor box is defined by $max(p_o * p_c)$. An anchor box will be considered as "containing an object" if the overall score is larger than the threshold, which is usually set to be 0.5 [13, 15].

In YOLOv3, the model will divide an image with width W and height H into 3 scales: W/32 x H/32, W/16 x H/16 and W/8 x H/8. In each scale, there are 3 anchor boxes is used, a total of 9 different anchor boxes is used in the whole model [13, 15].

Resnet

ResNet can train the model deeper. The formulation of $\mathcal{F}(x) + x$ can be realized by feedforward neural networks with "shortcut connections". Shortcut connections are the arrow skipping one or more layers. In the following figure(Figure 12), the shortcut performs identity mapping, and the outputs of the shortcut are added to the outputs of the stacked layers (two weight layers). Identity shortcut connections do not need extra parameters or computational complexity. The entire network can be trained end-to-end by Stochastic Gradient Descent (SGD) with backpropagation [8]. Since the residual block would not add any computational complexity, the training phase needs less time to achieve the required deep level while using ResNet.

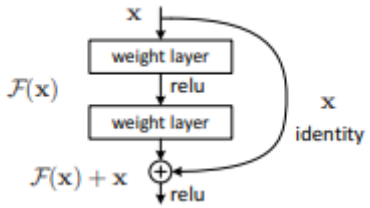

Figure 8. Residual network learning: a building block. [8]

CUITE model

Convolutional Universal Text Information Extractor (CUTIE) deals with problems of the key information extraction by applying convolutional deep learning model on the gridded texts. CUTIE uses the convolutional neural network (CNN) based network structure and involve the semantic features in a designed fashion. [18]

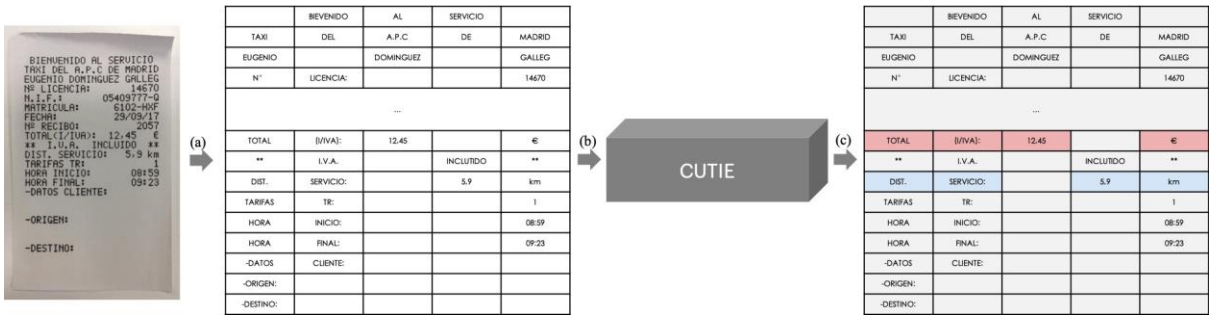

Figure 9. The structure of CUTIE model and Grid position mapping [18]

CUITE first generates input grid data for CNN by scanning input image processed in OCR engine for acquiring texts and their positions. In Figure 8, it shows the structure of the process. Grid Posit

ional Mapping is going to (a) map texts from the original scanned image to the target grid which w ould be as (b) input for the convolutional neural network in CUTIE model and then (c) output extr acted key information.

The mapping method would get the minimum bounding boxes around i-th interested text got in O CR engine from two restricted corner coordinates $(x^i{}_{left}, y^i{}_{top})$and $(x^i{}_{right}, y^i{}_{bottom})$. The ma pping would also find the center point $(c^i{}_x, c^i{}_y)$of the bounding box as a reference position in ord er to avoid problems of overlapped bounding and increase the accuracy of actual relative position. Consider the target grid size be (N,M). The mapping position of texts in the grid is calculated as

$$c_x^i = N \cdot \frac{x_{left} + \frac{(x_{right} - x_{left})}{2}}{w}$$

$$r_y^i = N \cdot \frac{y_{top} + \frac{(y_{bottom} - y_{top})}{2}}{h}$$

CUTIE, which is a CNN based model, learns to generate label of each text in the grid input throug h exploring the spatial and semantic features of texts in grids. CUTIE model contains two parts wh ich are CUTIE-A and CUTIE-B. Both part conducts semantical meaning encoding process with w ord embedding layer which is added a dropout to enhance the generalization of CUTIE model. CU TIE model uses cross entropy loss function to compare the predicted token class grid and the grou nd truth grid. CUTIE-A is a high capacity CNN that connects multi-resolution features without los ing high-resolution features to generate rich representations. CUTIE-B is a convolutional network which is constructed with a single backbone network but employs atrous convolution, which can e nlarge the field of view of filters to incorporate larger context, to capture long distance connection s.

2 Methodology

2.1 Design

2.1.1 Overall Design

In this project, we will turn the problem into 3 different sub-problems: text detection, text recogniti on, and text classification. In this report, we will mainly discuss the text detection and text recogni tion section.

First, an image of the receipt will be sent to the text detection model. The text detection model will detect and localize the text instances, and output the bounding box coordinate of all text instances.

Then, we use the bounding box to crop out the text instances in the image and send the cropped im ages to the text recognition model. The text recognition model will output the text instances in the format of string corresponding to each image.

Finally, we will use a text classification model to determine the class of the text instances. For exa mple, “2345 1111” will likely be a phone number and “$123” will be the amount. And the key information of the receipt will be output with the JSON format.

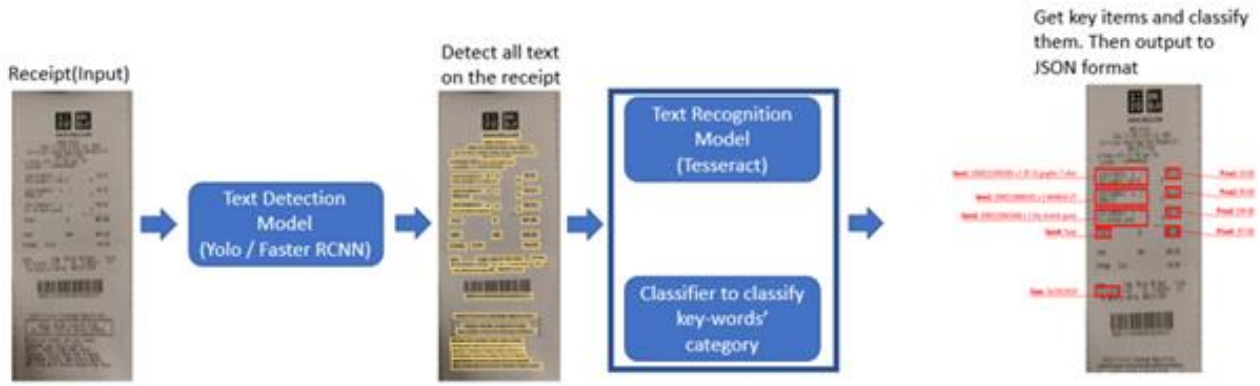

Figure 10. The design diagram of our model

While in recent research, researchers have created an end-to-end text recognition model that integr

ates both text detection model and text recognition model into a single network [12]. But due to th e limitation of the resources, we cannot fit a large model into the GPU. Hence we decided to use t he two-step model in the text recognition process and build the two models separately. This also gi ves us the advantage of parallelizing the training phase.

2.1.2 Text detection model

For text detection, we will reduce the problem to a general object detection problem by treating ea ch text as an object. After conducting the research, we decided to use YOLOv3 instead of Faster-R CNN as the object detection model. In some testing, YOLOv3 shows good results in terms of accu racy compared to Faster R-CNN but performs significantly faster than it [14]. Considering the limi tation of computational power we have, YOLOv3 would be a better choice for us.

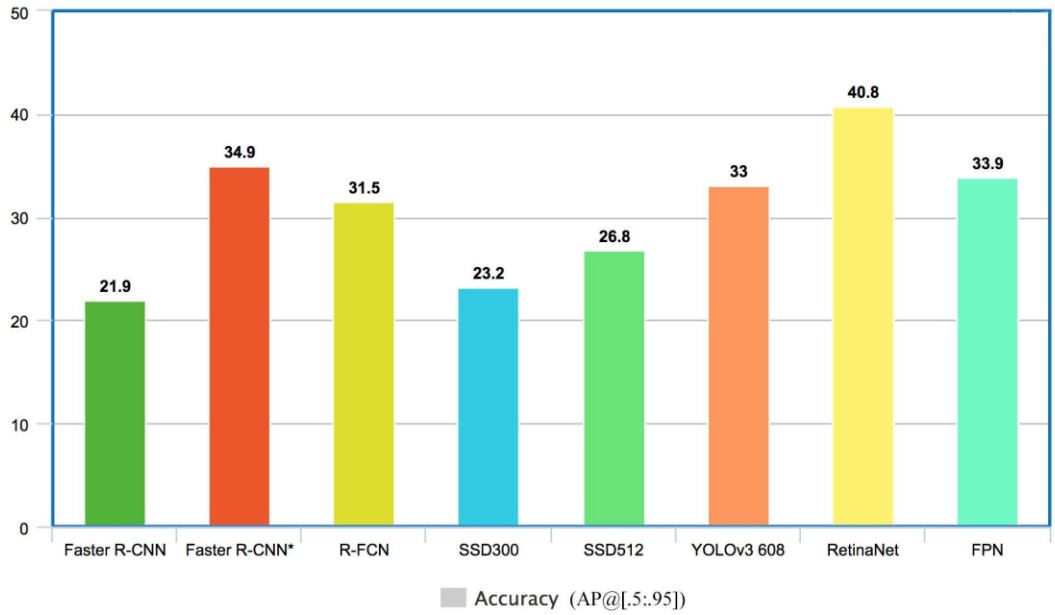

Figure 11. Accuracy of different models on MSCOCO dataset [14]

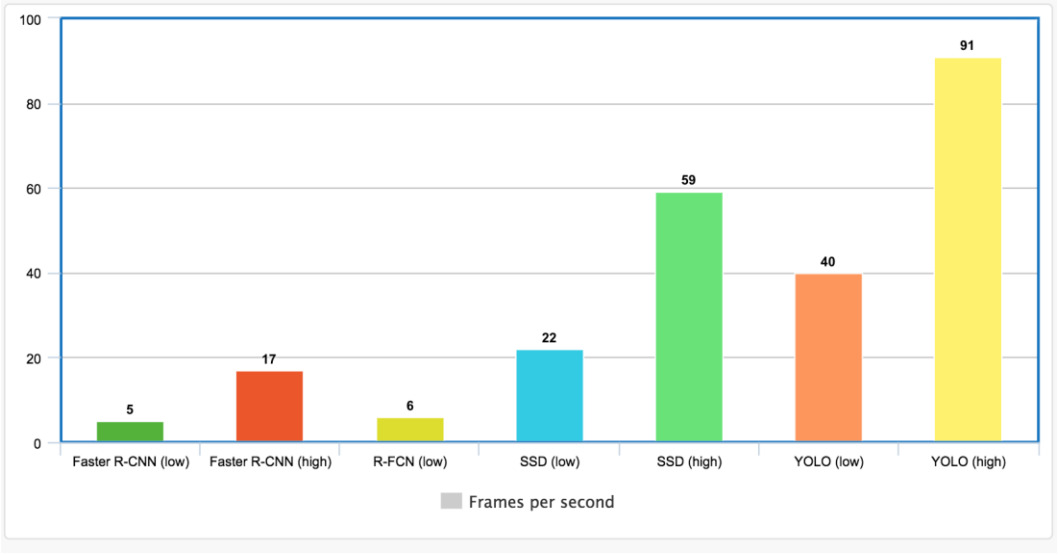

Figure 12. Frames per second achieved by different models [14]

In YOLOv3 paper, there are 3 anchors boxes at each scale, a total of 9 anchor boxes for general ob ject detection base on the COCO dataset [13, 15]. For the SynthText dataset, we plot the distributi on of the bounding box dimension as following, where x-axis and y-axis represent the normalized width and height respectively:

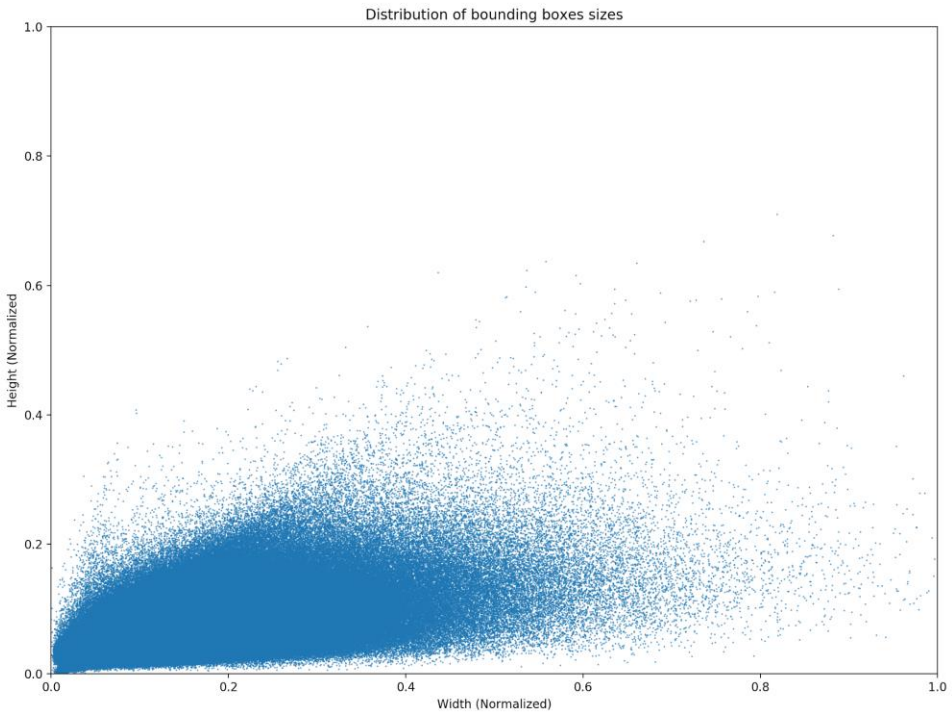

Figure 13. Bounding box dimension distribution

From our observation, the width of the bounding box and longer than the height of the bounding box in general. We would apply K-mean clustering to the dataset to obtain the size of the anchors with different K values. Using the calculated K anchors with different K values, we plot the average IOU versus K graph.

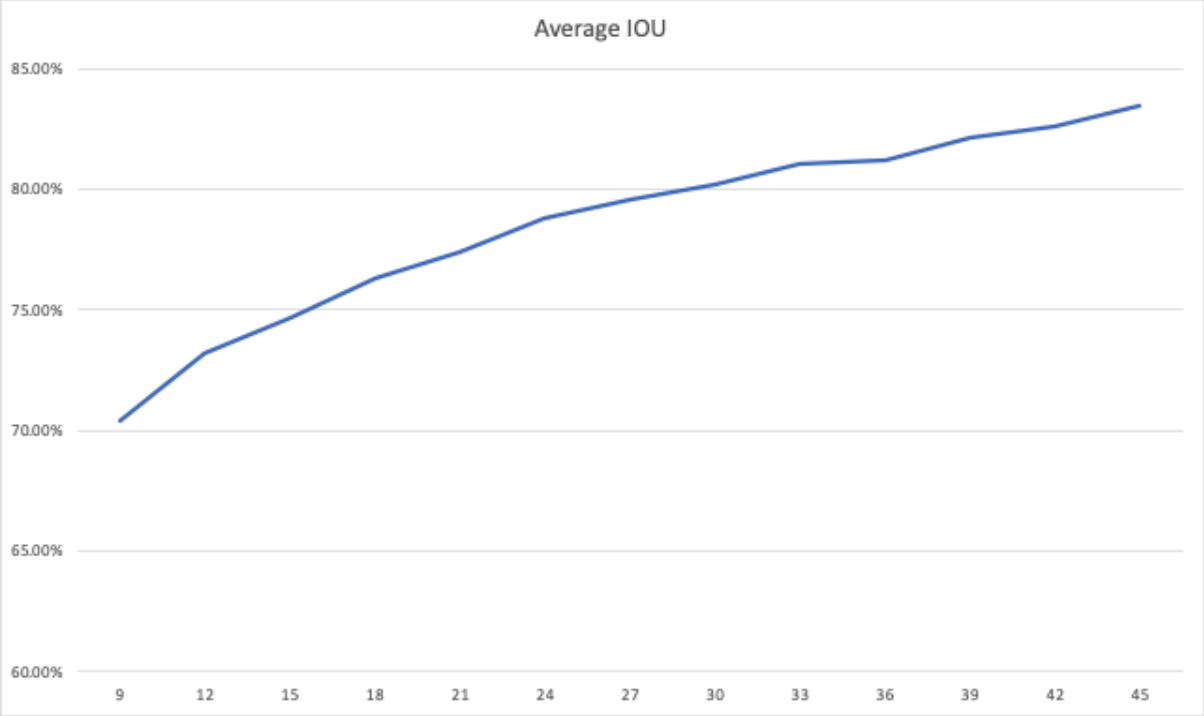

Figure 14. Average IOU versus K

We choose K=30 for the K-mean clustering which can obtain 80% of average IOU. As the result, we generate 30 anchors for YOLOV3, with 10 anchors at each scale.

In text detection, there is only one class of the object needs to be detected. While we can use the objectness score as the only classification indicator to determine the existence of an object inside an anchor box, we decided to use binary classification with two classes, "text" and "non-text", in the model. The total score of the anchor box will be calculated as $p_o * p_{text}$. This structure will give us good expandability in coding. If we want to integrate the classification task inside the text detection model, only a few lines of code are required.

2.1.3 Text recognition model

The Text recognition model uses Convolutional Neural Network (CNN) and Recurrent Neural Net works (RNN) to do tasks of feature extraction of images, and then feeds tensors into Bidirectional Long Short-Term Memory (Bi-LSTM) networks for improving the accuracy. Finally, the model w ould calculate loss by using Connectionist temporal classification (CTC) loss.

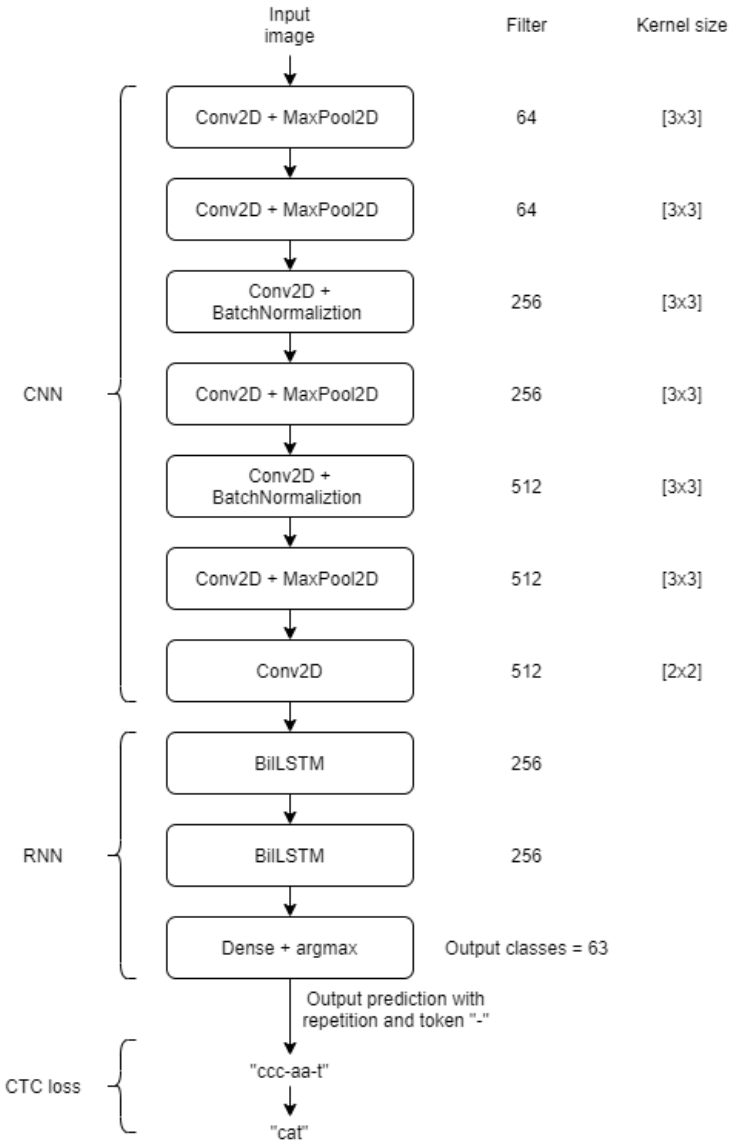

Figure 15. The design of the text recognition model.

CNN part would do features extraction of input image data by converting input to tensor data. RNN part can learn faster than one-directional approach by feeding the learning algorithm with the original data once from the beginning to the end and once from end to beginning in forward and backward LSTM layers. Connectionist Temporal Classification (CTC) loss is used as it tackles the time-consumption problem for annotating dataset on character-level and the problem of a single character spanning multiple horizontal positions, for example, to recognize “to” and “too”. A CTC network has a softmax output layer with one more unit than there are labels in L. The activations of the first |L| units are interpreted as the probabilities of observing the corresponding labels at particular times [5]. So we need to feed the output matrix of the Neural network and the corresponding ground-truth text (correct label) to the CTC loss function, and we also need to define converter function with features of decoding and encoding for converting text into integer label and removing duplicate characters and special character respectively, like “-” or blank space, from the prediction output of the text recognition model.

2.1.4 Information extraction model

Information extraction model for receipt information extraction would implement two methods of side-by-side comparison: Named Entity Recognition (NER) with sequence tagging and character e mbedding, and CUTIE model.

Character Embedding and Named Entity Recognition with sequence tagging

To begin with, we would define a process which convert raw text to a mathematical representatio n, called embedding. [19]

Character-level embedding is more general than word-level embedding. Word-level embedding wo uld convert words to integer tokens and a sentence to a list of integer token. With a word embeddi ng model, vector as a numerical representation can be generated and be fed into neural network. T heoretically, there are infinite English word combinations if we consider that English word is a seq uence of characters.
Although there is only a finite number of unique English words, there might be some missing data or words in current dataset of English word. Character embedding can avoid the problem that the model meets a new word that has never existed.

Consider there are only 52 English characters plus 20 special characters and 10 numbers. If we giv e each character a token, every English word can be represented as the sequence of integers. Then we would use char2vec, which is a character embedding model, to transform the sequence of integ ers into a single vector which is a numerical representation of a single word. If a word that does no t exist in the training dataset, we can still use character embedding to create a vector of this new w ord.

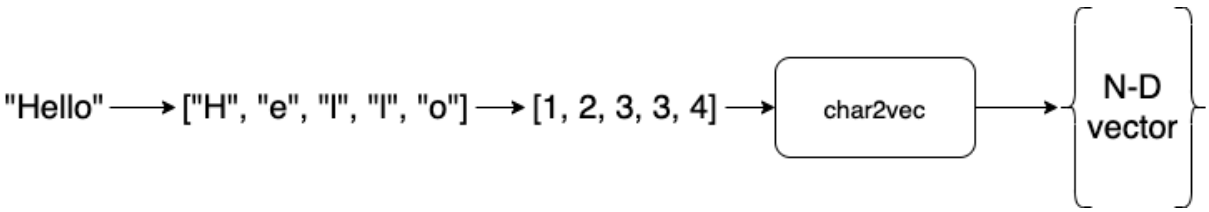

Figure 16. An example of character embedding which we give characters list [H,e,l,o] tokens in order of list [1,2,3,4]

Named entity recognition (NER) is a standalone tool for information extraction and also a role in n atural language processing (NLP) with a goal of recognizing mentions of rigid designators from te xt belonging to predefined semantic types. [20] In the information extraction task, we would defin

e 5 entities including merchant name, merchant address, transaction date, total, and a don’ t care c lass.

For each receipt, all words would be concatenated into a long sentence ordered from left to right a nd from top to bottom. With character embedding, each word in the sentence would be converted i nto a 150-d vector, and then form a sequence of vectors. After that, the data would be sent into the information extraction model. The model would output a sequence of class indexes that have the s ame length as the input sequence. Generally, the model aims to classify each word within a senten ce into different classes, or entities.

Sequence tagging includes part of speech tagging (POS), chunking, and named entity recognition (NER). Its output can be used for down streaming applications. Bidirectional LSTM Conditional R andom Field (Bi-LSTM-CRF) model can systematically compare the performance of models on N LP tagging data sets. And the model can be applied to NLP benchmark sequence tagging data sets. [20, 21] Therefore, for the information extraction model, we design our custom Bi-LSTM-CRF mo del, a simple recurrent model for sequence tagging.

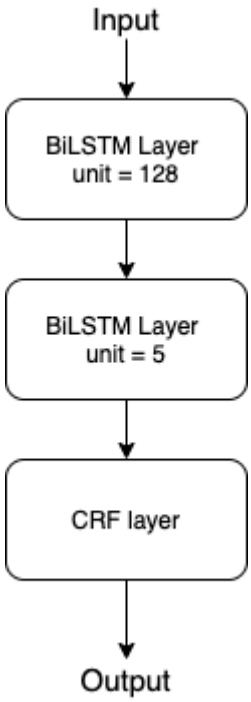

Figure 17. Structure of our BiLSTM-CRF model

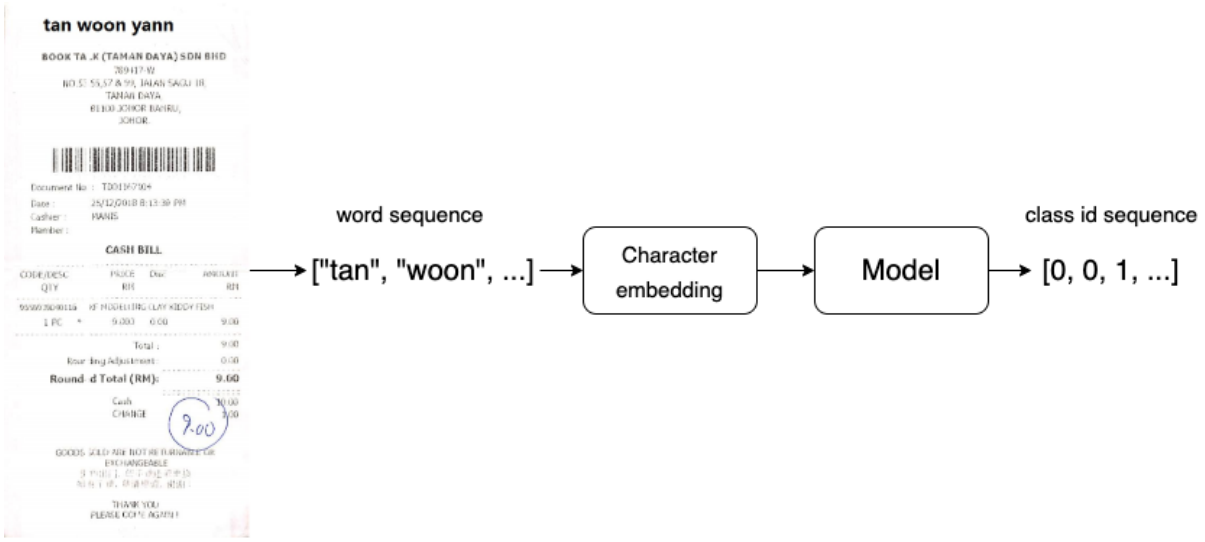

Figure 18. Overview of the sequence tagging method

CUTIE model

As mentioned in Session 1.3.2, CUTIE aims to extract information from a document in convolutio nal neural network (CNN) by preserving spatial relationship among texts in the input document wi th grid positional mapping method. [18] We would develop our model based on the CUTIE-B mod el, which is constructed with a single backbone network but employs atrous convolution, by addin g the convolutional block attention module.

Convolutional block attention module (CBAM) sequentially contains Channel Attention Module a nd Spatial Attention Module. Channel attention focuses on “what”’ is meaningful given an in put image. Spatial attention focuses on “where” is an informative part, which is complementary to the channel attention. Apart from the width and depth of networks, which affect the performanc e of CNN, Korea Advanced Institute of Science and Technology (KAIST) found that attention is a n important aspect of a architecture design. The attention mechanism can tell the network where to focus and improve the representation of interests. [22] It means that attention can make the model focus on important and related features.

For the size of the input data, we consider the grid size would be 64 x 64 and a total of 4096 cells for mapping the layout of the receipt. If the number of text instances that a receipt contains is betw een 200 to 400 on average, most of the cells would be either empty cells or irrelevant texts. Howe ver, CBAM model would help to focus on the more important text features but not empty cells or i rrelevant texts. The accuracy of key extraction could be improved.

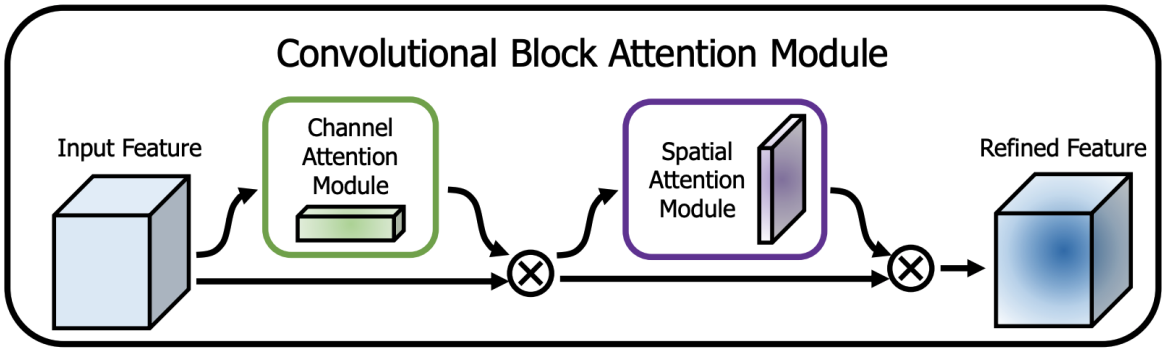

Figure 19. Structure of CBAM. [22]

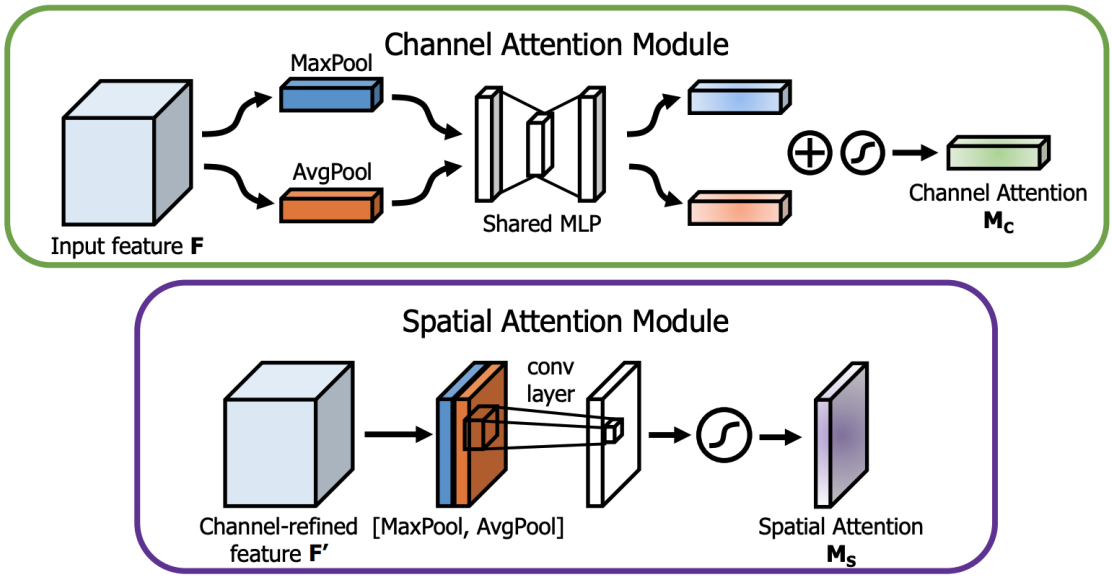

Figure 20. Details of Channel Attention Module
and Spatial Attention Module in CBAM. [22]

To generate the grid data, we first map the texts to the grid according to their relative position. The n each text within the cells will be converted into a 150-d vector by applying character embedding. The empty cells will be padded with 150-d vectors of zeros. Finally, an input tensor with shape 64 x 64 x 150 will be generated for the model.

Base on the structure of CUTIE-B model, we replace four convolutional layers in CUTIE-B model [18] to two ResBlock + CBAM blocks ,which is a implementation of CBAM as a block in ResNet, proposed by the CBAM paper [22]. The overall structures of ResBlock + CBAM model and our C UTIE model are shown in following figures:

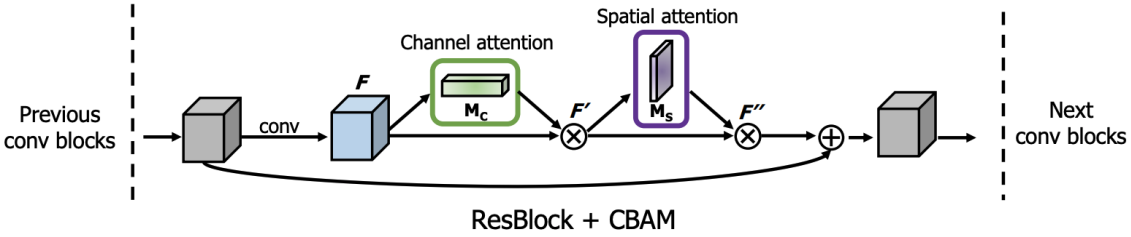

Figure 21. Structure of ResBlock + CBAM [22]

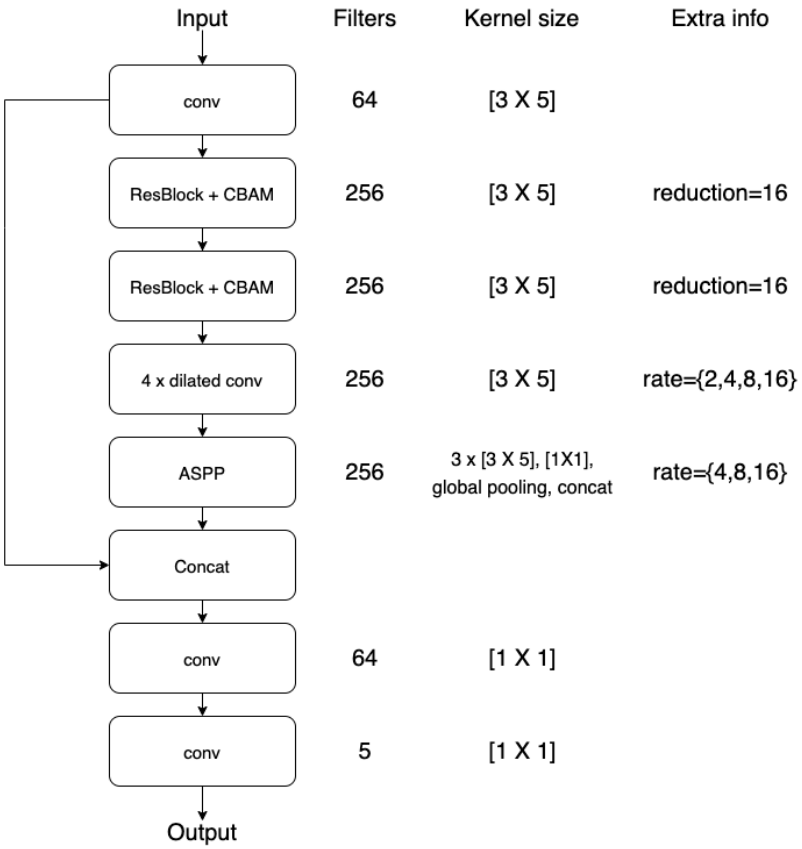

Figure 22. Structure of our custom CUTIE model by replacing a convolutional block to two ResBlock + CBAM blocks

2.2 Implementation

The deep learning model would be implemented with TensorFlow 2.0, a large deep learning frame work developed by Google [16].

2.3 Evaluation

Each model in the system would be evaluated separately according to different evaluation metrics.

For the text detection model, mean Average Precision (mAP) will be used as the evaluation metric. mAP is an evaluation metric for object detection which is widely adopted by object detection pape r including Faster R-CNN and YOLO [9, 10, 13, 17]. Text detection is a special case of object det ection, mAP is suitable for evaluating the performance of the model.

For the text recognition model, the output will be considered as “True” if the output is the same as the ground truth. Otherwise, it will be “False”. The accuracy of the model will be calculated based on the number of correct output divided by the total number of attempts.

3. Performance

3.1 Text detection model

YOLOv3 has been trained for 5 epochs with batch size 8, a total of 430k steps. At IOU threshold = 0.5, our YOLOv3 achieves 92.0% of mAP on the train set and 88.4% on the test set. At IOU thr eshold = 0.75, it achieves 84.6% of mAP on the train set and 80.0% on the test set.

After training, we create a prediction model and visualize the network output by plotting the boun ding box on the input receipt image. According to the output image, most of the texts on the receip t are being detected by our YOLOv3 model, while a small number of the texts are not being detect ed. The results and figures are as follows.

Table of the accuracy

mAP (%)	IOU=0.5	IOU=0.75
Train set	92.0%	88.4%
Test set	84.6%	80.0%

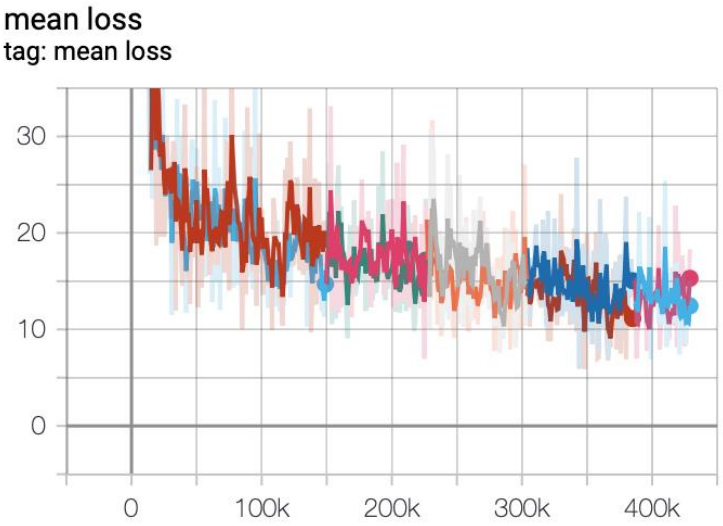

Figure 23. Mean loss graph of YOLOv3

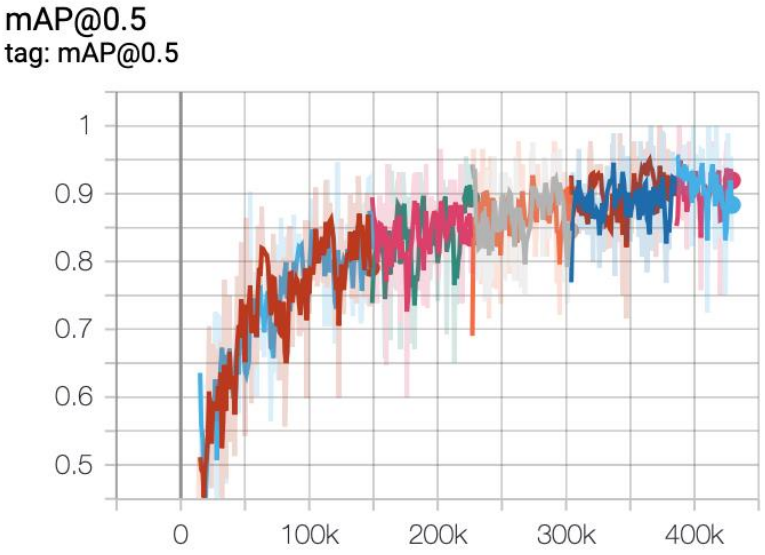

Figure 24. mAP graph of YOLOv3 where IOU threshold = 0.5

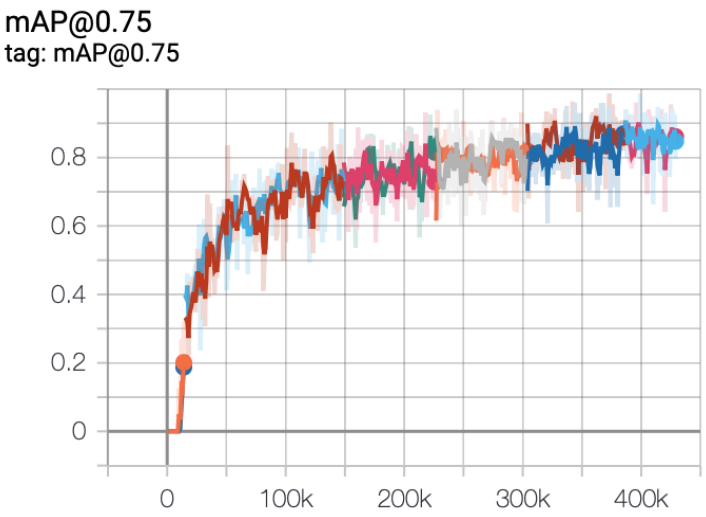

Figure 25. mAP graph of YOLOv3 where IOU threshold = 0.75

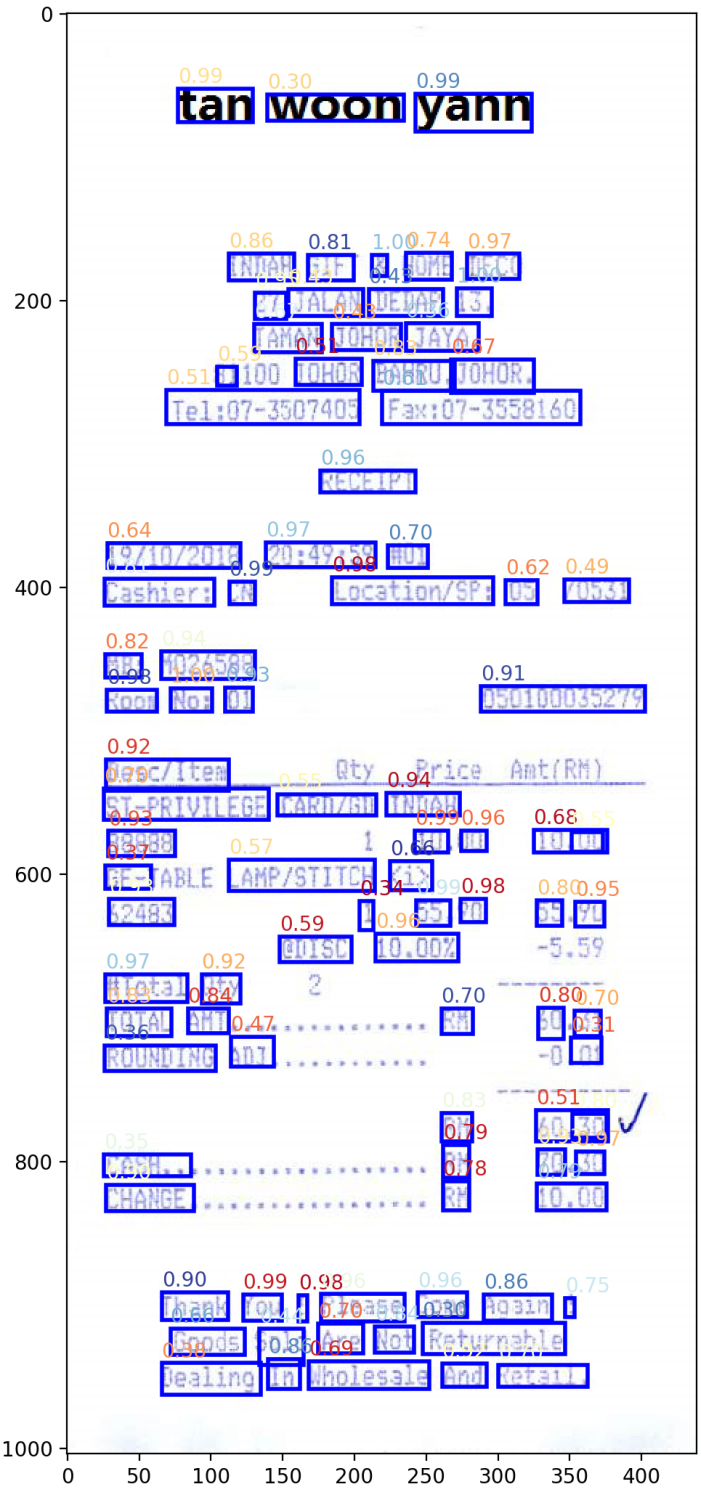

Figure 26. Bounding box prediction on the receipt image

3.2 Text recognition model

The training of text recognition would separate the dataset into 112884 batches which contain 64 images for each training batch. While $i \bmod 100 = 0$, where i is the training iteration, the model would use test dataset to calculate accuracy which would count total number of true positive cases in 124 test images. And the model would save the log data of accuracy and loss, as well as save a checkpoint if and only if new accuracy is higher than the original one. Since there are limitations of computer specification and network connection problems, the training process could not be continuous for a long period. Therefore, we could not get complete graphs of training data.The best results would be in the following table:

	Accuracy	Loss
Train set	92.19%	0.8
Test set	83.87%	

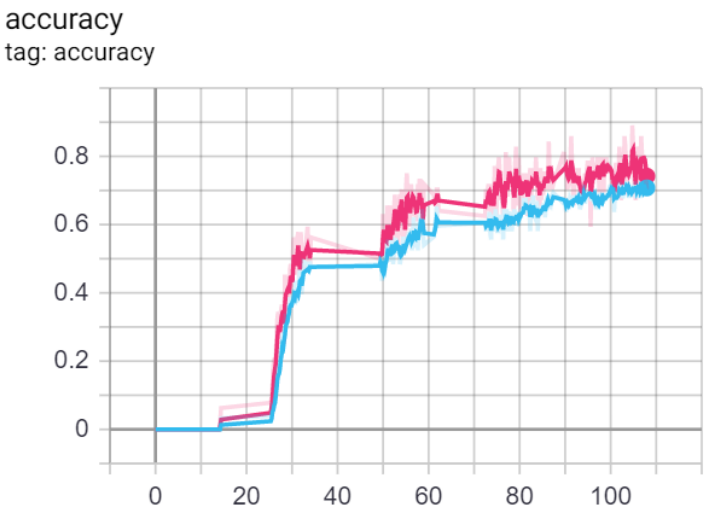

Figure 27. Relative trend of training and testing accuracy

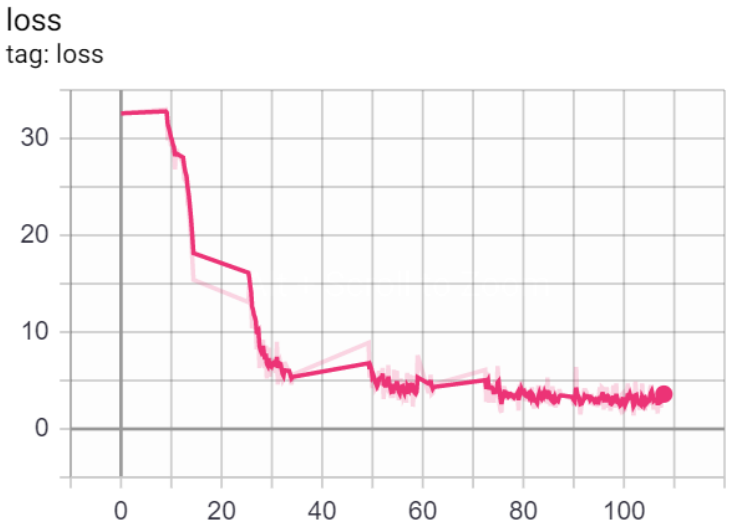

Figure 28. Relative trend of loss

3.3 Information extraction model

For the information extraction model, we have three models for the side-by-side comparison, including a BiLSTM-CRF model, the CUTIE model, and our custom CUTIE model. The three models are trained on the same dataset with around 1000 receipt images. 75% of the images will be split as the train set and the other 25% will be the test set.

Our custom CUTIE model achieves a mean f1-score of 99.6% on the train set and 74.7% on the test set. The CUTIE model achieves 98.3% on the train set and 71.1% on the test set. The BiLSTM-CRF model achieves 98.8% on the train set and 51.1% on the test set.

Table of accuracy

Accuracy (%)	BiLSTM-CRF model	CUTIE model	Our custom
Train set	98.8%	98.3%	99.6%
Test set	51.1%	71.1%	74.7%

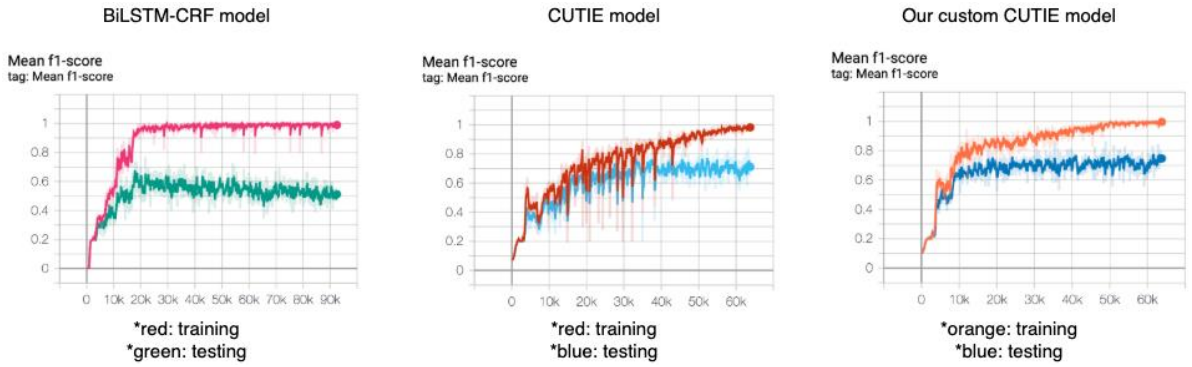

Figure 29. Accuracy of three models: BiLSTM-CRF model, CUTIE mode and our custom CUTIE model.

From the accuracy table and the graph, both three models achieve high accuracy on the train set bu t overfit on the test set. On the BiLSTM-CRF model, the model will reach high training accuracy q uickly, but the testing accuracy does not increase along with the training accuracy. After 20k steps, while the model reaches 95% of training accuracy, the testing accuracy stops at around 60%. On th e CUTIE model, the training accuracy continues to rise and reaches 98% of accuracy, while the tes ting accuracy stops at 70%. On our custom CUTIE model, the training accuracy rises quickly to 80 % and continues to reach 99% of accuracy. While the testing accuracy is being capped at 70% to 7 5%. According to the accuracy graph, the training accuracy of the CUTIE model drops suddenly d uring the training, result in unstable training performance. While the training accuracy of our custo m CUTIE model is more stable and reaches high accuracy faster than the original CUTIE model. I t shows that the CBAM has a positive effect on the CUTIE model.

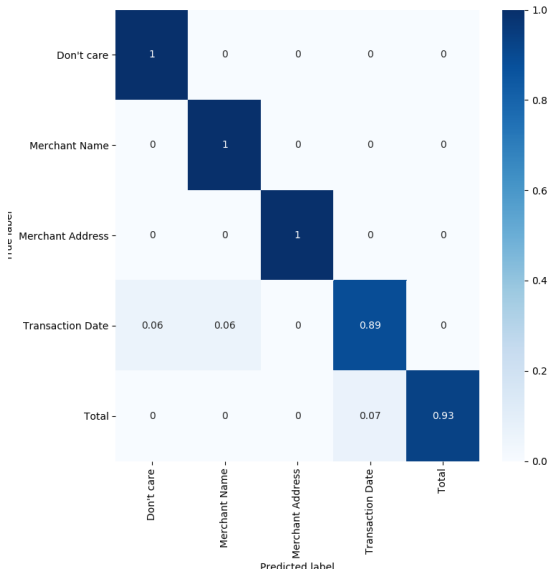

Figure 30. Confusion matrix of BiLSTM-CRF on the train set

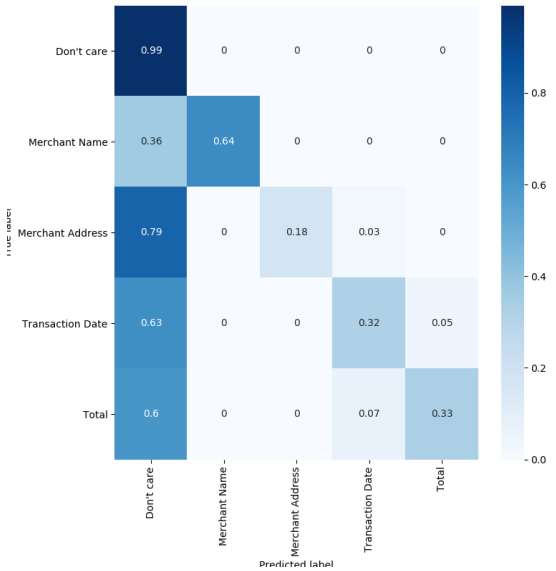

Figure 31. Confusion matrix of BiLSTM-CRF on the test set

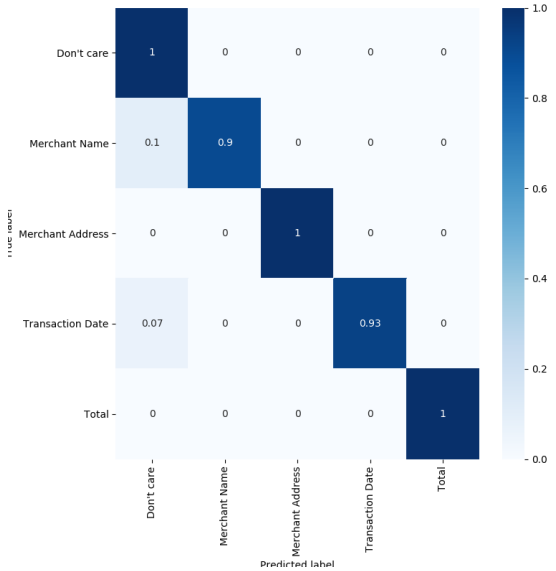

Figure 32. Confusion matrix of CUTIE on the train set

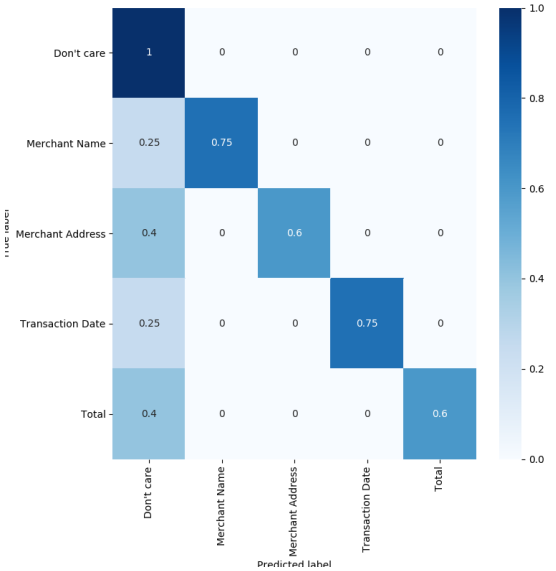

Figure 33. Confusion matrix of CUTIE on the test set

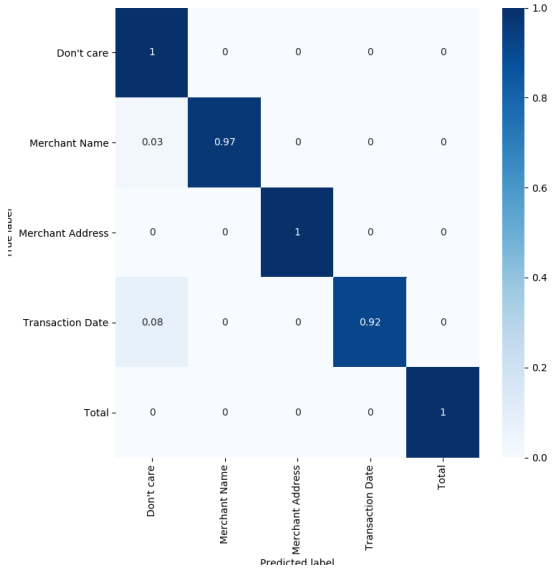

Figure 34. Confusion matrix of our custom CUTIE on the train set

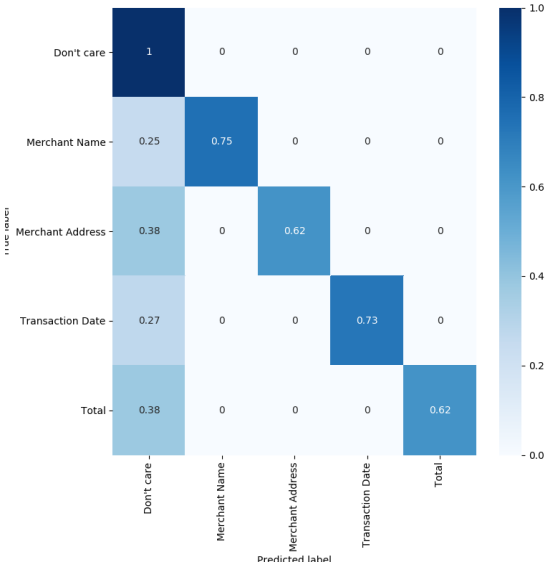

Figure 35. Confusion matrix of our custom CUTIE on the test set

4. Project Planning

4.1 Progress

	Choi	Wong
Summer	Discuss FYP topic, Read literature, and Study basic knowledge regarding Artificial intelligence and machine learning	
Oct	Finalize FYP topic, and Discuss with supervisor	
Oct - Nov	Study structures of text detection and recognition model and Read related literature, Collect dataset	
Nov - Dec	Build text detection model	Build text recognition model
Late Dec	Start training and tuning the models	
Jan - Mar	Build information extraction models and train all models	
Apr	Collect results and start writing report	

5. Overview

5.1 Conclusion

In this receipt recognition project, we divide the whole process into three tasks and tackle those thr ee tasks individually with different methods.

In the text detection section, we use state-of-the-art object detection model YOLOv3 as our text de tector. After training on the SynthText in the Wild dataset for over one week, we achieve over 80 % of mAP on the test set. According to the example output on the receipt image, most of the texts can be detected by our model, while a small number of texts are not being recognized. While consi dering the accuracy graph of the YOLOv3 model, where the accuracy keeps increasing at the end of the training phase, we believe that the model will perform better if we trained the model for a lo nger period of time. However, due to the lack of hardware resources, which we will mention in sec tion 5.2, this is the best of what we can achieve.

In the text recognition section, we use CRNN model ,which is a combination of CNN model and R NN model, with CTC loss. The training process is extremely time consuming since the dataset is l arge and the limitation of computer specification and network connection. The computer cannot us e Graphics Processing Units for training and the application of GPU usage on Google Cloud Platfo rm has failed. The training process has taken over one month to achieve over 90% of accuracy in t rain set and over 80% of accuracy in test set. While doing the training, case sensitivity would be c onsidered in outputs and accuracy calculation. We believe that the accuracy would be higher witho ut considering case sensitivity.

In the information extraction section, we compare the performance of the three different models, in cluding the BiLSTM-CRF model, the CUTIE model, and our custom CUTIE model. Our model ge nerally performs better than the BiLSTM-CRF model and the CUTIE model. However, all three m odels overfit on the test set, as the test accuracies are significantly worse than the training accurac y. We believe the reason for overfitting is due to the lack of data. We train the models on 700 rece ipt images. Compared with other deep learning training datasets which usually have more than 10k of samples, the number of images is small. The model does not see enough samples to generalize o ver different types of receipts.

5.2 Difficulties and Limitation

Time cost

Projects regarding Artificial Intelligence or machine learning are time-consuming for the training phase. The dataset should be numerous and as "real-world" as possible which means we need to collect real datasets related to the project. In our case, we should collect all types of receipts with different features (font style, color of the font, language and positions of key items) if we would like to obtain the best accuracy. However, it is unrealistic to do so since it should be impossible to collect full datasets of all types of receipts. Therefore, we need a very large size of datasets to train our model for holding good accuracy. The training phase would be expensive and slow.

Artificial Intelligence and machine learning are still fresh and developing. We need to learn deeper rather than keep on the fundamental level. Although there are many resources, like source code, papers, and articles, it is hard to choose the most appropriate model, layers for our case of scanning receipts.

Limited hardware resource

As there is a high requirement of GPU in the training phase and high specification GPU is extremely expensive. It is time-consuming if we use our only graphic card to process training. For example, training YOLOv3 with image size 416x416 and batch size 12 requires 16GB of VRAM in the GPU, which can be only found on high-end enterprise GPU. The training phase would be held on cloud computing service, like Google Colab, or using CSE GPU account.

Lack of dataset

For the information extraction model, we collect 1000 receipt images for training. While a dataset with 1000 images is quite small in the field of deep learning. While this is the maximum number of free receipt images we can collect. There are some databases that contain a huge amount of receipt images for the user to purchase, but we cannot afford the expensive price tag. Therefore, we decide to train the model with those 1000 free images.

6. References

[1] Veit, A., Matera, T., Neumann, L., Matas, J. and Belongie, S. (2016). “COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images”. [online] arXiv.org. Available at: https://arxiv.org/abs/1601.07140.

[2] Ankush Gupta and Andrea Vedaldi and Andrew Zisserman. (2016). “Synthetic Data for Text Localisation in Natural Images”. IEEE Conference on Computer Vision and Pattern Recognition.

[3] Max Jaderberg and Andrea Vedaldi and Andrew Zisserman. (2014). “Deep Features for Text Spotting”. European Conference on Computer Vision

[4] V. Sivakumar, A. Gordo, M. Paluri. “Rosetta: Understanding text in images and videos with machine learning”. Facebook Engineering AI RESEARCH. [Online]. Available: https://engineering.fb.com/ai-research/rosetta-understanding-text-in-images-and-videos-with-machine-learning/

[5] Alex Graves, Santiago Fern´andez, Faustino Gomez, J¨urgen Schmidhuber “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks”. [Online]. Available: https://www.cs.toronto.edu/~graves/icml_2006.pdf

[6] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun. “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices”. [Online]. Available: https://arxiv.org/abs/1707.01083

[7] A. Ng. “C4W2L03 Resnets - Case Study. Residual Network (Resnet)”. [Online]. Available: https://www.youtube.com/watch?v=ZILIbUvp5lk

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun . “Deep Residual Learning for Image Recognition”. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun , “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. [Online]. Available: https://arxiv.org/abs/1506.01497

[10] Ross Girshick. “Fast R-CNN”. [Online]. Available:https://arxiv.org/abs/1504.08083

[11] Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation”. [Online]. Available: https://arxiv.org/abs/1311.2524

[12] Qin, S., Bissacco, A., Raptis, M., Fujii, Y. and Xiao, Y. (2019). “Towards Unconstrained End-to-End Text Spotting”. [online] arXiv.org. Available at: https://arxiv.org/abs/1908.09231 [Accessed 17 Dec. 2019].

[13] Redmon, J. and Farhadi, A. (2018). “YOLOv3: An Incremental Improvement”. [online] arXiv.org. Available at: https://arxiv.org/abs/1804.02767 [Accessed 20 Dec. 2019].

[14] Hui, J. (2018). “Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and…”. [online] Medium. Available at: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359 [Accessed 20 Dec. 2019].

[15] Medium. (2019). YOLO v3 theory explained. [online] Available at: https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193 [Accessed 21 Dec. 2019].

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[17] Hui, J. (2018). mAP (mean Average Precision) for Object Detection. [online] Medium. Available at: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173 [Accessed 21 Dec. 2019].

[18] Zhao, Xiaohui, Niu, Endi, Wu, Zhuo, Wang, and Xiaoguang, “CUTIE: Learning to Understand Documents with Convolutional Universal Text Information Extractor,” arXiv.org, 20-Jun-2019. [Online]. Available: https://arxiv.org/abs/1903.12363. [Accessed: 20-Apr-2020].

[19] M. Antonio, “Word Embedding, Character Embedding and Contextual Embedding in BiDAF-an Illustrated Guide,” Medium, 05-Sep-2019. [Online]. Available: https://towardsdatascience.com/the-definitive-guide-to-bidaf-part-2-word-embedding-character-embedding-and-contextual-c151fc4f05bb. [Accessed: 20-Apr-2020].

[20] Li, Jing, Sun, Li, and Chenliang, “A Survey on Deep Learning for Named Entity Recognition,” arXiv.org, 18-Mar-2020. [Online]. Available: https://arxiv.org/abs/1812.09449. [Accessed: 20-Apr-2020].

[21] Huang, Xu, Wei, Yu, and Kai, “Bidirectional LSTM-CRF Models for Sequence Tagging,” arXiv.org, 09-Aug-2015. [Online]. Available: https://arxiv.org/abs/1508.01991. [Accessed: 20-Apr-2020].

[22] Park, Jongchan, Lee, Joon-Young, and Kweon, “CBAM: Convolutional Block Attention Module,” arXiv.org, 18-Jul-2018. [Online]. Available: https://arxiv.org/abs/1807.06521. [Accessed: 20-Apr-2020].