EE628 Final Project Report

Xiangyang Xu
Department of Electronic Engineering
Stevens Institute of Technology
Hoboken, NJ, 07030
xxu46@stevens.edu

Abstract

This project is to use CNN and MNIST dataset to achieve digital classification. At the same time, this project uses some features of target images. So that without using object detection algorithms such as YOLO, it is possible to find all digits in the image and know digit location.

1 Introduction

Currently, people use some tech like object-detection for finding. Target detection is a great way to find specific targets. However, many times, people face the lack of a good enough dataset to train a model with higher accuracy. At the same time, people do not have enough time to make a good enough dataset, and making a large enough dataset is a quite time-consuming work.
In some special cases, I can use the various features in the image to get the target we want without using object detection technology.

This project faces this unfavorable situation. There is not enough time to build tens of thousands of images and labeled dataset, and open good source datasets are not available online. Based on the digits I write in One Note, I found some ways to make it possible to get the desired pictures without using object detection.

Using the MNIST dataset, I successfully trained a CNN model of handwritten digit classification with high accuracy. Successfully split each digit in a picture into 28 * 28 pixels pictures, making it suitable for CNN. Teacher may use this to easily score students' handwritten math work.

In this report, Part 2 is about CNN part of my project and Part 3 is about image procession of my project.

2 CNN

Convolutional Neural Network (CNN) is a feed-forward neural network. Its artificial neurons can respond to some of the surrounding cells in the coverage area. has excellent performance for large-scale image processing.

A convolutional neural network consists of one or more convolutional layers and a fully connected layer at the top (corresponding to a classic neural network). It also includes associated weights and a pooling layer. This structure enables the convolutional neural network to utilize the two-dimensional structure of the input data. Compared with other deep learning structures, convolutional neural networks can give better results in image and speech recognition. This model can also be trained using back propagation algorithms. Compared with other deep, feedforward

neural networks, convolutional neural networks need to consider fewer parameters, making it an attractive deep learning structure. [1]

2.1 Model

I used a CNN model with the following layers, conv layer 1, conv layer 2, max pooling layer, dropout layer 1, flatten layer, dense layer 1, dropout layer 2 and dense layer 2.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1179776
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Model Summary

2.2 Train

This model is trained by Adam optimizer, uses crossentropy to calculate its loss. I set train 10 epoch. If model accuracy and loss is good enough, I will not try more epoch.

Table 1: Epoch, Loss and Accuracy

Epoch	Loss	Accuracy
1	0.1881	94.34%
2	0.0777	97.68%
3	0.0618	98.18%
4	0.0509	98.52%
5	0.0429	98.66%
6	0.0352	98.92%
7	0.0342	99.00%
8	0.0294	99.10%
9	0.0255	99.17%
10	0.0221	99.25%

Currently, loss is small enough and accuracy is more than 99%. I think this accuracy and loss is acceptable.

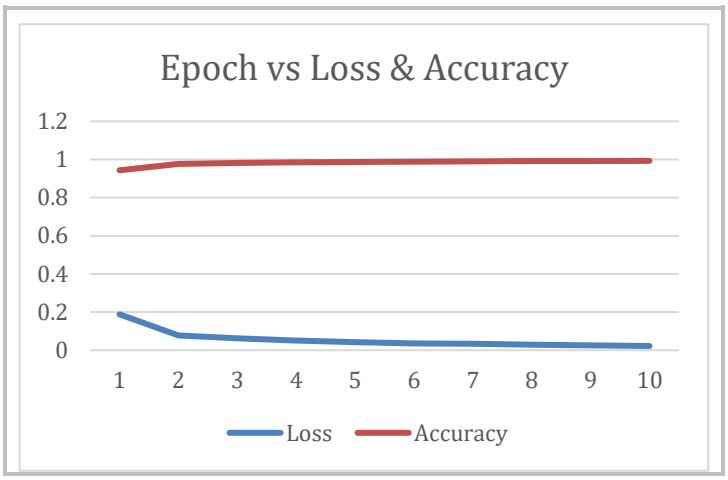

Figure 2: Epoch, Loss and Accuracy

2.3 Model Prediction

After training this model, I evaluate this model with test data label and its prediction. This model get loss 0.0298 and accuracy 99.22% with test data.

Then I tested the first twenty data, and their predicted values is the same as the label.

n	test_y	predict_
1	7	7
2	2	2
3	1	1
4	0	0
5	4	4
6	1	1
7	4	4
8	9	9
9	5	5
10	9	9
11	0	0
12	6	6
13	9	9
14	0	0
15	1	1
16	5	5
17	9	9
18	7	7
19	3	3
20	4	4

Figure 3: Predict Value and Label

3 Image Processing

Generally speaking, to find different digits in the same picture requires a lot of data to train an object detection model. In this project, through the use of some unique characteristics of worksheets and digits, the location of digits can also be determined. Then take a screenshot of the digit and process it into a picture of 28 * 28 pixels. It can be trained by the model

based on the MNIST dataset.

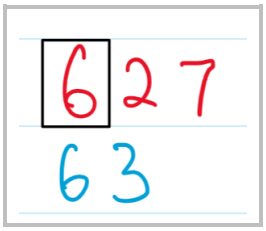

Figure 4: Digit Box

Currently, As Figure 4 show, the job of image procession is to know the left, right, top and bottom coordinate. Then I just need to do some image procession to make the digit image fit 28*28 pixels and other technique requirement of model trained by MNIST datasets.

3.1 Split Image based on Y Axis

As the sheet is line by line, I can detect the coordinate of blue line and use this as top and bottom of a digit.

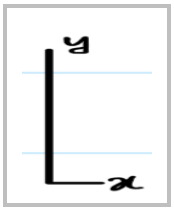

Figure 5: Y Axis Split

As Figure 5 shown, in the first pixel of the first column, there should be no digit, and every pixel are not white may be a line. By calculate with the RGB of [0,i] pixel, numbers between 0~255, I can judge it's a white pixel or not. If R, G and B are 255 or very close to 255, I can think it's white or almost white. Otherwise, the pixel is not white, then I think that it is the y-axis coordinate of the possible line.

At this time there is another problem, multiple adjacent pixels represent a line. So I must merge the Y coordinates of these pixels so that they represent the same line.

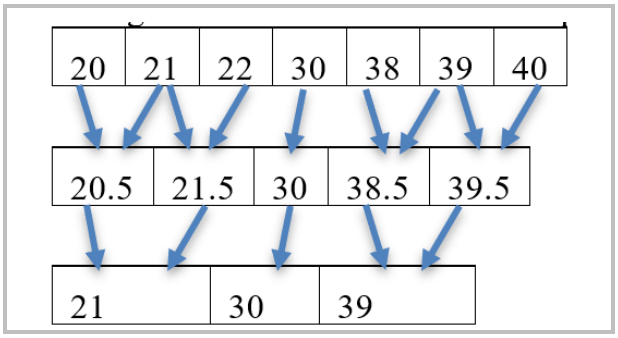

Figure 6: Merge Thick Lane Algorithm

Figure 6 shows an algorithm that can merge thick lines into an accurate, thin line. The numbers in the figure represent the y coordinate of non-white pixels. If, I think they are a thick line, then calculate their average as the new line. After iterating several times, we can change all the thick lines into thin lines with a width of 1.

Figure 7: Sample Y Axis Split

Now, I know the y-axis coordinates of each blue line, and I can split a whole picture into multiple pictures based on this. Figure 7 is a part of Y-Axis split of a test image.

I create list L. Each element of L is the new list L_i. I save each new image as the first element in L_i.

3.2 Split Image based on X Axis

Now the image has been divided into small images. I just need to find the X-coordinate of the number. Similarly, I also use the RGB of the calculated pixels to distinguish whether the image is white. I built a matrix with the same pixels as the image. All non-white pixels are recorded as 1, and all white pixels or nearly white pixels are recorded as 0.

Blue line	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
White Or Non-White							1	1	1	1	1								
						1	1				1	1							
						1				1	1								
									1	1									
								1	1										
							1	1											
							1				1	1							
						1	1	1	1	1	1	1							
Blue line	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SUM	2	2	2	2	2	5	7	6	6	6	7	6	2	2	2	2	2	2	2

Figure 8: Sample Pixels Array of '2' Image

As shown in Figure 8, by calculating the sum of 1 in each column, I can find out how many pixels in this row are non-white. I set the start and end positions of values continuously greater than the lowest value 2 as $start_j$ and end_j. The second element of a line in list L_i will record it. Second element works as $[[start_1, end_1], [start_2, end_2], \ldots, [start_j, end_j]]$.

Then I can get left and right coordinate of a digit by $start_j$ and end_j. Now, it is possible to accurately cut the picture of a line into several digits.

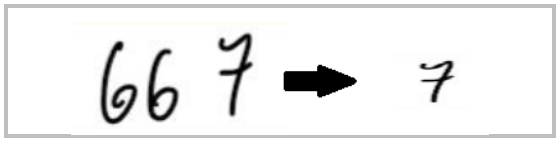

Figure 9: Bad Sample X Axis Split

Finally, we need to turn the picture into 28 * 28 pixels.

We cannot simply reshape digital images. As shown in figure 9, this will cause the digital image to be severely distorted, and digits are too flat. This eventually leads to a very large error in the number judged by the CNN model. There is a 30~50% probability that the digit will be judged wrongly, when I test some image at beginning.

Then I used a new method. I firstly created a white background of the same size n * n as the large value n of height h and width w. Then the original image is pasted to the center of the new n * n background. Then reshape the n * n image to a 28 * 28 image.

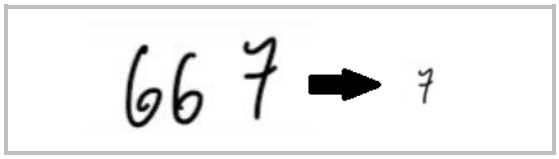

Figure 10: Good Sample X Axis Split

Through the new method, the accuracy of CNN model judgment is greatly improved. Figure 10 shown a sample image that will be judged by CNN model.

I store every new digit picture in third element of L_i. L_{i3} is a list as well.

3.3 Image Procession

After going through the above steps, I only need to do some final processing to judge the new pictures using the CNN model. I make a same n*n size numpy array to restore the standard data from RBG data for CNN model. New data = 1 - (R + G + B)/ (3*255).

Then make this file to shape (1, 28, 28, 1). Now, it's good for CNN model

3.4 Result

Firstly, upload the pre-trained model. Use pre-trained CNN model to predict the result of the digit image. I restore all prediction to a new numpy array, which shape is same as L_{i3}. This will help me calculate the final solution. For example, I need to make a list [5, 9, 7] for calculating number 597.

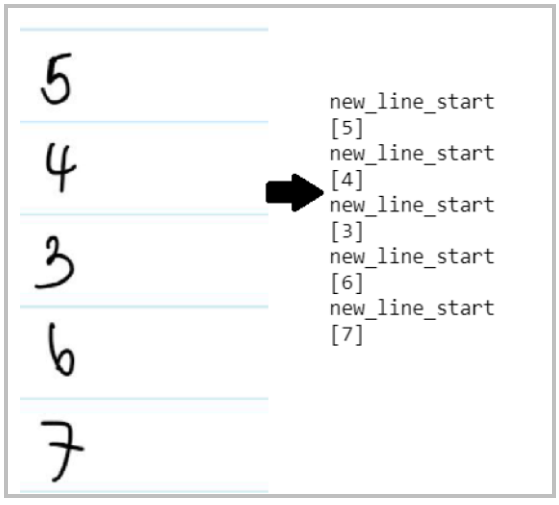

Figure 11: Sample test result with test image

As Figure 11 shown, currently, all function seems good now. I just need to calculate the final

number with the list of number. For example, I calculate [5, 9, 7] in the following way, $5*10^2+ 9*10^1 +7*10^0 = 597$.

Finally, we just judge the difference between teacher solution and student solution and get student's score.

4 Conclusion

This project is a special use of the features of a specific image, so that the image classification model and picture positioning algorithm can play the role of object detection. At the same time, the training image classification model is faster and more accurate than the training object detection model.

Due to the lack of sufficient test images obtained by scanning, this project will have many special situations that have not been taken into account and may cause bugs.

References

[1] http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

.