Vehicle Number Plate Recognition Using Optical Character Recognition

Gui-Han Go
Department of Computer Science
Georgetown University
Washington, District of Columbia
gg626@georgetown.edu

Chia-Hsuan Hsieh
Department of Computer Science
Georgetown University
Washington, District of Columbia
ch1165@georgetown.edu

Kevin Kyunggeun Jang
Department of Computer Science
Georgetown University
Washington, District of Columbia
kj460@georgetown.edu

***Abstract*—Optical Character Recognition (OCR) is one of the key technologies in Automatic Number-Plate Recognition (ANPR). In this study, our team is building a machine learning model that implements OCR system to complete the character recognition tasks on the vehicles number plate images. More specifically, our group is applying the concept of both a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) over Recurrent Neural Network (RNN) to build a model that can accurately recognize characters in the image of a number plate.**

I. Introduction

As years pass by, things that were done manually with human's hands gets automated and digitized by computers. As a result, the importance and significance of technology increases and often times, it enormously benefits people's daily lives. Optical Character Recognition (OCR), [1] the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image, is one well-known example. Through OCR, manual works like data entry for business documents (e.g. check, passport, invoice, bank statement and receipt), automatic number plate recognition, and many others are now automated by computers. This automation is hugely successful in processing a large amount of works in a very short period of time, but never be more accurate than the works that are carefully done by humans. In other words, the quantity and speed of works is now guaranteed, but the quality of works is not yet perfect. As an instance, Automatic Number-Plate Recognition (ANPR) still have the following difficulties: [2] blurry images caused by a motion blur, poor lighting, an object obscuring part of the plate, and many others, which still requires humans attention to complete the recognition process. Therefore, further researches and studies are essential to overcome that flaws.

In this project, our group mainly focuses on applying OCR in recognizing the characters in the image of vehicle number plates and targeting to get the accuracy over 90%. Contrary to the previous approaches, our team is implementing the concept of both Convolutional Neural Network (CNN) with two convolution layers and an average pooling operation, and Long Short-Term Memory (LSTM) over Recurrent Neural Network (RNN).

II. Related Work

- Character Recognition in Automatic Vehicle License Plate Recognition by Anuj Kumar [3] Unlike our project, this study used videos of a moving car as dataset. Their approach on data processing was first convert the video into frames, select key frames, and extract the plate image from the image of a vehicle. This image will be passed to their character recognition system to output the plate numbers. In their training process, Artificial Neural Network (ANN) was used to extract the pixels in the number plate image. Then, a standard back-propagation learning algorithm is used for training and testing the model. In conclusion, they could not solve the ambiguity problem between two similar looking characters such as o and 0, I and 1, B and 8, C and G.
- Number Plate Recognition Using OCR Technique by Er. Kavneet Kaur and Vijay Kumar Banga [4] This study used the template matching approach to recognize the characters in a plate. Instead of building a model using a neural network, this group simply selected the characters from the image by cropping the characters and used their noise removal technique to increase the readability of the characters. They concluded that their approach was not effective especially when there is noise in image and on recognizing characters printed with different font types.

III. Datasets

For this project, our group used the artificially generated dataset that are very similar to the real world vehicle number plates. This dataset were downloaded from Supervisely website. The dataset contains total of 22,764 files, where half of the files are in JSON format and another half in PNG format. Out of 22,764 files, our group decided to use half of the files (11,382 files) to train the model and another half (11,382 files) to test our model.

A. File Types

- JSON : This file contains the information on the label (plate number), size (width and height) and tag (train or

test) of the given number plate. The above information will be used as the attributes for the given number plate.
- PNG : This file is the artificially generated image of the given number plate.

B. Data Structure

There are three classes created to store the dataset.

- TrainTestDataSet : This class contains the DataSet object for both training and testing datasets. The build_train_test_dataset function in this class can be called to build the datasets.
- DataSet : This class contains the list of DataElement object for a single dataset (training or testing). The build_dataset function can be called to read and parse the given data files and create and store the DataElement object for each data file.
- DataElement : This class contains the variables for each attribute obtained from a single data file. There are total of five variables in each DataElement object, which are:

1) label : label (plate number) of a given number plate
2) height : measured height of a given number plate
3) width : measured width of a given number plate
4) label_length : number of characters in a given number plate
5) img : parsed image file

C. Data Cleanliness

The cleanliness of the data were measured to avoid using any data with flaws. The following values were used to measure the cleanliness.

- Data Redundancy : There may exist redundant data in both training and testing dataset. This can result giving more weights to this redundant data while training the model. Therefore, our group should investigate, detect and remove redundant data from the combined dataset. This operation will improve data quality and make data analysis process more accurate.
- Data Missing : There may exist data that are missing key attribute values. Our group will remove those data to improve the overall quality of the dataset.
- Noise Data : Our group put an effort to reduce data with noise attribute values from collected data as much as possible to really increase the accuracy and preciseness on our predictions.

Due to the above reasons, our group will create functions to clean the combined dataset to maximize the overall quality of our analyses. However, surprisingly, the dataset were already clean enough. Further analysis on this will be explained below.

D. Data Cleaning Process Logic

The following are the steps our group followed to resolve and measure the cleanliness issue on the dataset:

1) In the process of reading and parsing each data file, verify the cleanliness of this data using the valid_json function in DataSet class using the below steps.
2) Check if the given data has any missing attribute values (label and size of the given number plate).
3) Check if the given data has any noisy attribute values. In our case, the given data is considered as noisy if any of the following conditions are not satisfied: the number of characters in a label must be eight, and the size of the plate must be 152 by 34 (width by height).
4) Check if the given data already exists in the dataset.
5) If the given data violates any of the above steps from step 2) to 4), our group dropped this data from the dataset.

IV. Methods

Our model combines the structure of both CNN and RNN. Our team first used CNN to extract the most important features and used LSTM over the RNN model. In CNN, we performed the convolution twice followed by a (2, 2) Average Pooling for each convolution. After this, we reshaped the layer to make it become 38 x 128 x 1 layer structure. We then create a fully connected layer, a dense layer, for the input of the RNN. The shape of the input of RNN is 38 x 16. For simplification, we named the two layer nodes in the layer just after the 38 x 16 layer to be 1-1 and 1-2, and the 4 layer nodes after the layer containing 1-1 and 1-2 to be 2-1, 2-2, 2-3, and 2-4. Also, the two layer nodes after that are named 3-1 and 3-2. In RNN, we do forward LSTM and backward LSTM to the input of RNN, so we have layer node 1-1 and 1-2 after the input layer of RNN. After this process, we repeat the same procedures again. We do forward LSTM and backward LSTM to both layer node 1-1 and layer node 1-2. Now, we have four 38 x 128 layers in our RNN model. We then add layer nodes 2-1 and 2-2 to get layer node 3-1 as a result, and add layer nodes 2-3 and 2-4 which outputs layer node 3-2. Then, we concatenate layer node 3-1 and layer node 3-2 to come up with a 38 x 256 layer and dense the layer to achieve a 38 x 23 layer. We then apply softmax function to the layer and use CTC Loss function at the end. We will explain further on why we dense the layer to achieve 38 x 23 layer later in this paper.

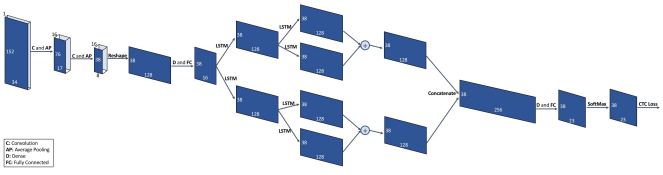

Fig. 1. The structure (CNN + RNN) of the model

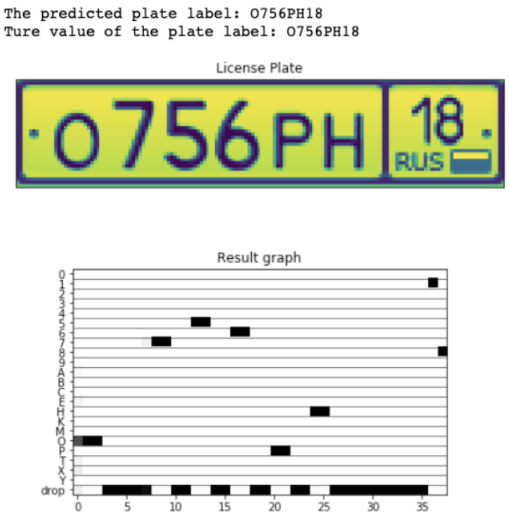

Fig. 2. The probability distribution of the corresponding license plate

V. Results

In order to shorten the total execution time, our group decided to only select 1/5 of random training data to train our model. However, once we came up with the better design of our model, we trained and tested the model with the complete set of dataset. After training our model using with the half (11,382) of the total dataset, we tested the model with the other half (11,382) of the total dataset. Out of 11,382 number plates we tested, our model correctly recognized and identified all the characters in 11,344 number plates. The accuracy for this experiment was 99.67%.

VI. Discussion of Results

There are four different cases of character misclassification in our model as a result of testing 11,382 datasets.

1) M-H case (33 cases)

Actual	Predict	Actual	Predict
M811CO23	H811CO23	M621AK64	H621AK64
M485PX01	H485PX01	M400XX13	H400XX13
M611KK30	H611KK30	M487EC72	H487EC72
M420AC63	H420AC63	M428BK14	H428BK14
M446EK01	H446EK01	M488EP04	H488EP04
M416XE79	H416XE79	M407KE25	H407KE25
M444OA24	H444OA24	M618YP57	H618YP57
M418TE15	H418TE15	M814OP22	H814OP22
M821CH75	H821CH75	M418KO16	H418KO16
M487MX74	H487MX74	M471HM35	H471HM35
M445XB95	H445XB95	M414EK29	H414EK29
M481XM98	H481XM98	M815PP48	H815PP48
M441BM58	H441BM58	M816HC94	H816HC94
M470AC71	H470AC71	M471EM77	H471EM77
M815AC17	H815AC17	M810CB08	H810CB08
M474KB70	H474KB70	M815YH35	H815YH35
M444HX75	H444HX75		

Fig. 3. Misclassified character M to H

2) M-K case (2 cases)

Actual	Predict
M152KY86	K152KY86
M155EP17	K155EP17

Fig. 4. Misclassified character M to K

3) O-C case (2 cases)

Actual	Predict
O611AK84	C611AK84
O611HC07	C611HC07

Fig. 5. Misclassified character O to C

4) Etc case (1 case)

Actual	Predict
M152KY86	K152KY86
M155EP17	K155EP17

Fig. 6. Other misclassified cases

For most of the misclassified license plate containing M in the beginning, our model misclassified it to the character H or K. In addition, the model also misclassified some of the character O to C.

In the works of Supervisely, they used two max pooling after each CNN layer. After that, they used two of 512 x 32 GRU layers, and added them up to achieve a 512 x 32 layer. Then, they performed GRU again to this layer to come up with two of 512 x 32 GRU layers. This time, they used concatenate method to make their two layers to acquire a 1024 x 32 layer. The structure that they used looks like a 1-2-1-2-1 structure.

In our model, we created our own methods to come up with different model design. First, we used an average pooling in the CNN process. In RNN, we first made the dense layer to achieve two of 128 x 38 LSTM layers, and perform LSTM to each one of them to achieve four of 128 x 38 LSTM layer. Our team then added the first and second layer (layer_add_1) and added the third and fourth layer (layer_add_2). Next, we concatenated the two layers (layer_add_1 and layer_add_2).

Our structure looks like a 1-2-4-2-1 structure. This model resulted a 99.67% accuracy with the filters equal to 64, 128, and 256 in RNN. In addition, our group also performed the 1-2-1-2-1 LSTM RNN experiment. As a result, the model output an accuracy equivalent to 97%. The results of 1-2-1-2-1 and 1-2-4-2-1 structures were very similar. However, when we modified our pooling layer by changing the number of layers to one and pooling size to (4, 4), two structures resulted slightly different outcomes. In 1-2-1-2-1 LSTM structure, if we set the pooling size to (4, 4) and perform the average pooling once, then the accuracy now became 86%. However, if we set the pooling size to (2, 2), and perform pooling twice, the accuracy became 97%. In 1-2-4-2-1 LSTM structure, if we set the pooling size to (4,4) and perform the average pooling once, the accuracy became 97%. However, if we set the pooling size to (2,2), and we perform pooling twice, the accuracy became 99%.

Layer Structure		1-2-1-2-1	1-2-1-2-1	1-2-4-2-1	1-2-4-2-1	1-2-4-2-1	1-2-4-2-1
CNN	filters	16	16	16	16	16	16
	Pooling times	2	1	2	2	1	2
	Pooling size	(2, 2)	(4, 4)	(2, 2)	(2, 2)	(4, 4)	(2, 2)
RNN (LSTM)	filters	128	128	64	128	128	256
Accuracy		97%	86%	99%	99%	97%	99%

Fig. 7. The comparison table for different pooling and different layer structure

VII. Conclusions

The most challenging and important part on building our model was making pairs of the LSTM in RNN process, since one of them is forward and the other one is backward. Our team then added them up or performed concatenation. We would have gotten a very different result with a low accuracy if we did not insert the backward LSTM layer to our model. As our future work, we will put more effort on simplifying our model design to enhance the overall performance on training the model especially to reduce to total execution time. In addition, we want to test our model with various kinds of RNN models and other activation functions to verify whether we can come up with the model that can increase the accuracy on recognizing the characters on the number plate.

References

[1] “Optical character recognition”, Wikipedia, The Free ENcyclopedia. Wikipedia, The Free Encyclopedia, 28 Nov. 2018. Web. 30 Nov. 2018.

[2] “Automatic number-plate recognition”, Wikipedia, The Free ENcyclopedia. Wikipedia, The Free Encyclopedia, 21 Nov. 2018. Web. 03 Dec. 2018.

[3] Anuj Kumar, “Character Recognition in Automatic Vehicle License Plate Recognition”, in International Journal of Advanced Research in Computer Science and Software Engineering. 2016, vol. 6.

[4] Er. Kavneet Kaur, Vijay Kumar Banga, “Number Plate Recognition Using OCR Technique”, in IJRET: International Journal of Research in Engineering and Technology. 2013, vol. 2.

[5] “Latest Deep Learning OCR with Keras and Supervisely in 15 minutes”, Hacker Noon. Supervisely.ly, 2 Nov 2017. Web. 28 Nov 2018.