|  |
| --- |
| **CMPT 414 Project: License Plates Recognition** |

**Yiming ZhangChongyuyuan**

301354482 301354567

yza440@sfu.ca chongyuy@sfu.ca

**1 Introduction**

The project is proposed with the fundamental purpose of using Convolutional Neural Network (CNN) to identify license plates. License Plates Recognition contains many steps, like image preprocessing, image segmentation, etc. But this project is only focused on using CNN to identify single license plate letter or digit. After training, this model can correctly recognize about 93% of the license plate characters it has not seen. If we can get more and better data sets, the accuracy will further increases.

**2 Data Processing**

We used Chinese license plates as the training set, because in China, all provinces use a unified license plate style, and we can collect them more easily. A Chinese license plate consists of exactly one Chinese character at most left position and six capital English letters or digits (Figure 1.1). In this project, we just ignore the most left position and only deal with English letter A-Z and 0-9, which is more generalized.



**Figure 1.1**

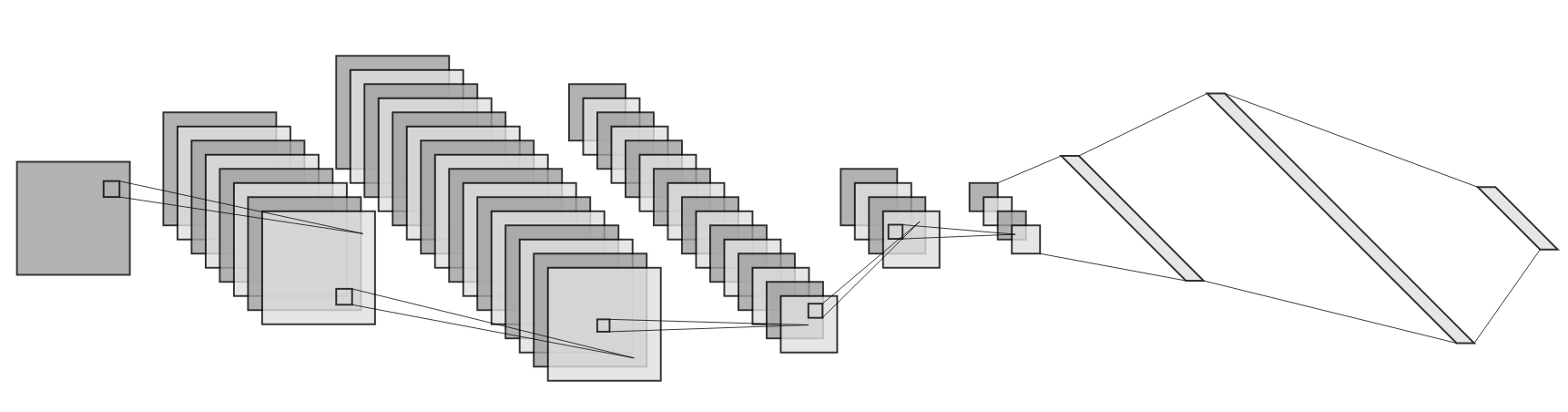
We found a dataset containing more than 5000 Chinese license plates images, which were taken by camera from different angles (<https://download.csdn.net/download/king900704/10659970>). First, each image was adjusted to 20x20 with median filtering, and then transferred to binary image (Figure 1.2). Finally, those images are separated into three parts: training set, validation set and test set, which were used as our model’s input.



**Figure 1.2**

**3 Network Structure**

We use Google’s Tensorflow module to implement this network in Python. The hidden layers of the entire network include three convolution layers and two maximum pooling layers, followed by a fully connected layer. The input layer has 400 nodes as our images has 20x20 pixels. And then the output layer has 34 nodes (output 0-9 and A-Z, there is no letter ‘O’ and ‘I’ in Chinese license plates). Detailed network structure is showed in Figure 2.



Convolution

(3x3 kernel)

Input Image

20x20

Convolution

(3x3 kernel)

Convolution

(3x3 kernel)

Max-pooling

(2x2 kernel)

Max-pooling

(2x2 kernel)

Flatten

(1x200)

Fully-connected

(1x512)

Output

(1x34)

64 feature maps

Stride = 1

64 feature maps

Stride = 2

ReLU Activation

32 feature maps

Stride = 1

**8**

8 feature maps

Stride=1

8 feature maps

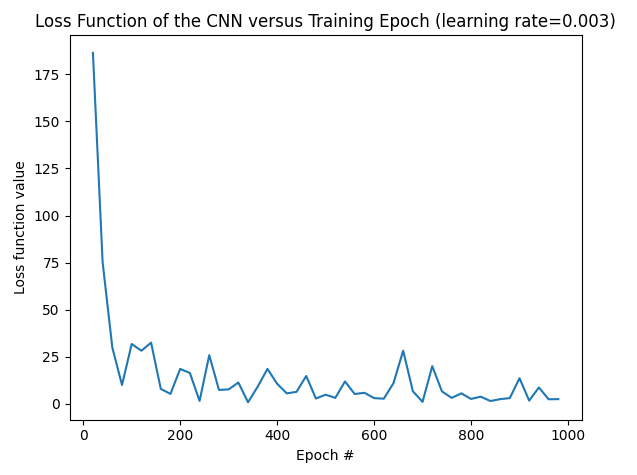
Stride=2

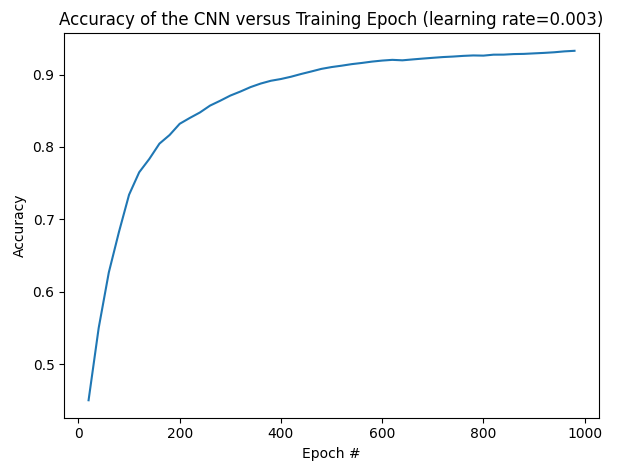
Soft-Max

**Figure 2**

**4 Network Training & Testing**

We found that too small or too large learning rate will cause the loss function to converge towards the local minimum too slowly. After several experiments, we use 0.003 as a suitable learning rate. To avoid the over-fitting problem that the model performs perfect on training images but poor on test images, we randomly dropped some neurons at fully connected layers. The accuracy on validation set and the loss function value are recorded every twenty epochs (Figure 3).





**Figure 3**

**5 Conclusion**

After 1000 epochs’ training, loss function is close to zero and the accuracy calculated based on validation set of the model is 92.96%. Due to the limited number and quality of training sets, e.g, the image cropping of the our dataset is not very good, some characters are slanted, the accuracy of this model can be further improved, we can expect higher accuracy in a lager and better training sets. From this project, we learned how to design, build and train a complete neural network and apply it to practical problems. And also, all codes are written by ourselves, which is also a good coding practice.

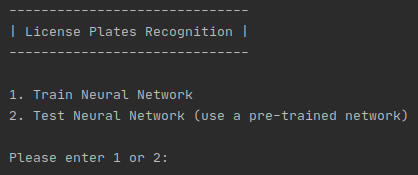
**6 How to run the code**

This project is written in python3.6, please make sure you have installed:

- Pillow

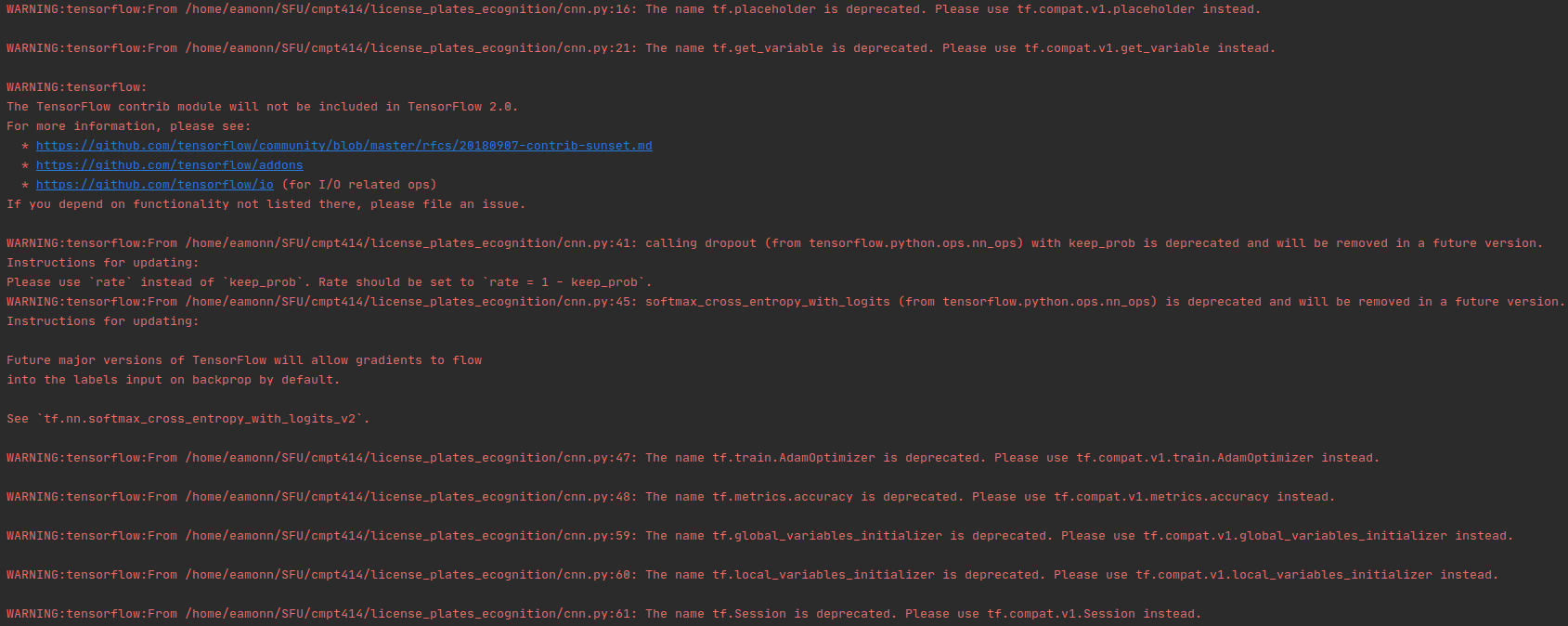
- matplotlib

- tensorflow 1.15.2 (Do not use any version > 2.0 as some functions are deprecated in newer versions)

The main function is in cnn.py, after running it, a menu will pop up (Figure 4)

**Figure 4**

You can choose either to train a new CNN or use a pre-trained CNN which is stored in ***./license\_plate\_cnn*** to test new images. During the training or testing process, you may see some warning info like:



Please ignore them, they are not error, the main reason is that we are not using the latest version of tensorflow.