ENGN4528/6528 Term Project Report – Deep License plate reader

Dongtao Yu (u5930159)
Taku Ueki (u5934839)

1. Introduction

Automatic vehicle identification has been around for many years. In particularly, Automatic License Plate Recognition (ALPR) is a method of vehicle identification. It is used in many areas, such as traffic control, parking identification, etc. Although many algorithms and methods have been proposed and developed dramatically in the past decades, it is still an active research area that has not been conquered. ALPR is accomplished by a combination of many techniques; for example, character detection, image segmentation and recognition. Therefore, it is important to choose proper methods to create a system to recognize license plates automatically.

Today, research about deep learning and image recognition with Convolutional Neural Network (CNN) is also very active around the world. These technologies enable us to recognize and classify images real time more accurately than other image recognition techniques. Therefore, they are utilized for many applications such as image recognition and character recognition. Other recognition and detection methods are replaced by these technologies.

So, we developed an ALPR system by combining image segmentation method specifically connected component based algorithm and CNN based character recognition.

2. Related Work

Regarding license plate recognition, there are three themes; license plate detection, license plate segmentation, and character recognition.

a) License plate detection

Many license plate detection algorithms have been proposed during the past years. Those existing algorithms can be divided into 4 groups; edge based, color based, texture based, and character based algorithms [1]. Edge based algorithms find license plate region by detecting the region whose change of brightness is more remarkable and frequent. Color based algorithm detects license plate region from color difference between car body and license plate. Texture based algorithm detects the unconventional pixel intensity distribution to find license plate region. Character based algorithm assumes that license plates consist of string of character and detects them by finding characters in the image.

b) License plate segmentation

Many algorithms to recognize license plate also exist, they can be classified into 2 groups; projection based and connected component based algorithm. Projection based algorithms project binary license plate images horizontally to determine height and top bottom boundary of the characters and vertically to determine width and boundary between each character. Connected component based algorithms find connected components in the binary image by labeling connected pixels. To increase accuracy these two approaches can be combined [2].

Related to license plate segmentation, image binarization is also important and many algorithms are proposed. Especially an algorithm presented by Christian Wolf enables us to detect characters with high detection rate [3]. Wolf's algorithm set an arbitrary size of window and calculates a threshold T for the center pixel of the window with following formula.

$$T = m - k\alpha(m - M) \ , \ \alpha = 1 - \frac{s}{R} \ , \ R = \max(s)$$

Where m is local average value of window, k is fixed to 0.5, M is the minimum gray value of image, s is local variance of window.

c) Character recognition

To recognize license plate characters which are segmented by above algorithms template matching based and learning based have been proposed [1]. Template matching based method measures the similarity between character and template and recognizes character as the most similar template. Learning based method extracts some features such as image density and gradient from images and uses them to recognize the input character.

d) State-of-the-art research about license plate recognition

The success of deep neural networks in computer vision applications inspired many researches. A system recognizes license plate in a natural scene image by using neural networks was presented this year [4]. This system uses three neural networks. First one is 37-class CNN which detects the character in the input image. Second CNN is trained to classify inputted image from first CNN plate or non-plate. Finally, a recurrent neural network with long short-term memory is trained to recognize detected license plate characters. Sequential features extracted from license plate are labeled by this neural network.

e) CNN parameter updates

Gradient descent optimization (GDP) for CNN is also active area of research. Choosing proper optimizer is indispensable to get better parameter and faster. Adaptive Moment Estimation (Adam) is one of recommended GDP algorithms [5]. This algorithm was inspired by earlier GDP algorithms such as Momentum and Adadelta. Momentum is one of GDP algorithms this algorithm calculates and stores exponentially decaying average of past gradients. Momentum helps accelerate stochastic gradient descent in the relevant direction. After this Momentum was proposed, Adadelta was appeared to solve problems Momentum has. Adadelta's algorithm calculates and stores exponentially decaying average of past squared gradients. By doing this, Adadelta can prevent aggressive decreasing of learning rate in Momentum's algorithm. Adam's algorithm stores both exponentially decaying average of past gradients and squared gradients. Therefore, Adam helps stochastic gradient descent converge to proper value and faster.[6]

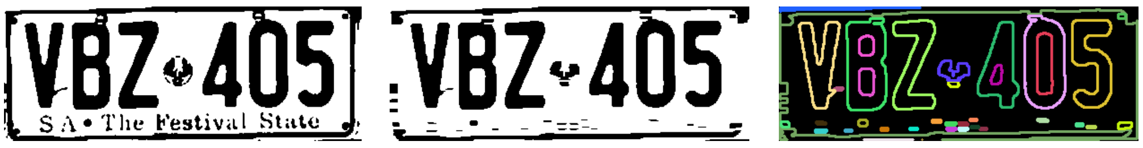
3. Terminology

Contour – a curve joining all the continuous points along the boundary, having same color or intensity.

4. Approach

The project is divided into two phases; the first phase is to correctly identify and segment each character from the license plate image, the second is to train a convolutional neural network in order to classify each segmented character in to 36 classes(A-Z and 0-9).

The first phase of the project was implemented using C/C++ in aid of OpenCV library. First the program takes a license plate image path as a command line arguments. It reads in the original images and converts it into a grayscale image. Then median filter and Gaussian filter with same kernel size 3 by 3 are applied to the grayscale image so that the Gaussian noise and salt and pepper noise are reduced. The program then binarizes the image by using Wolf's algorithm [3] instead of global thresholding techniques such as Otsu's algorithm [7] and mean thresholding. This algorithm slides a window step by step across the entire image and calculate an adaptive binarization threshold for each sliding window. Hence, different region of images gets a different threshold so that the image with varying illumination will get a better result. After binarization, the connect component analysis is applied in order to find every blobs. However, from Fig. 1(a), the character B and O are both connected to the image border, by doing a connected component analysis, the character B and O will be classified into the same blobs along with the border. It is an undesired situation which makes the program very difficult to distinguish both characters. Hence, a rectangular kernel size 7 by 1 dilation is applied to the image so that the bridge between characters and image border is removed Fig. 2(b). Then connected component analysis applied by using findContours function in OpenCV library. From Fig. 1(c), clearly, there are lots of blobs in the image and a method is proposed to extract useful blobs (characters) in the image.



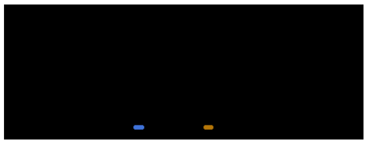
(a) (b) (c)

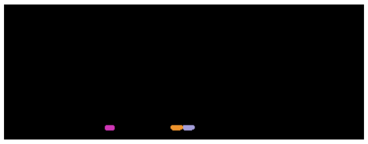
Fig. 1 (a) Image after Wolf's binarization. (b) Image after dilation. (c) contours after connected component analysis

The pseudo code of proposed method:

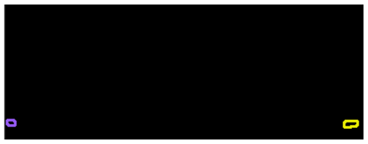
```
## initalize constant
MAX_WIDTH_RATIO = 2.8
HEIGHT_TH_RATIO = 0.1

groups = []

for i_contour in all contours:
  i_rect = BoundingRect of i_contour
  group = []
  append i_contour to group

  ## find the contours that has the similar height and
  ## has the similar y starting point and
  ## width ratio is smaller than threshold
  for j_contour in all contours:
    if i_contour != j_contour:
      j_rect = BoundingRect of j_contour
      h_threshold = (iRect.height + jRect.height) * HEIGHT_TH_RATIO

      if i_rect.width > j_rect.width:
        width_ratio = i_rect.width / j_rect.width
      else:
        width_ratio = j_rect.width > i_rect.width:

      if (abs(i_rect.y - j_rect.y) < h_threshold and
          abs(i_rect.height - j_rect.height) < h_threshold and
          width_ratio <= MAX_WIDTH_RATIO):
          append j_contour to group


  ## try to merge the group if has a single element in common
  already_merge = False
  for existing_group in groups:
    for element in group:
      if not already_merge:
        if element in existing_group:
          existing_group += group
          already_merge = True

  ## if not merged
  if not already_merge and size of group != 1:
    append group to groups
```

Result after applying the proposed method:

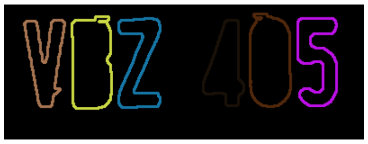

(a)


(b)

(c)


(d)

(e)

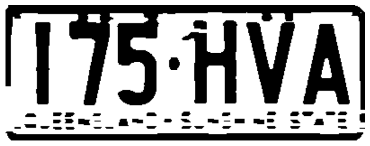


(f)

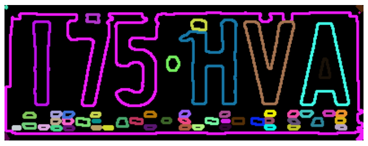
Fig. 2 groups of contours that returned by the method proposed

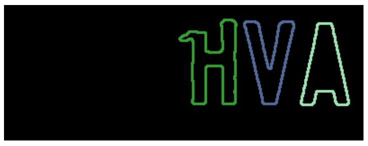
From Fig. 2, choosing a group that has the maximum average are will be the result. However, from Fig. 3(c), the result is not so great. Since the close morphology operation did not remove the bridge connecting image border and characters. Thus, in the connect component analysis the characters will be classified along with border. To overcome this issue, one assumption was made that at least one character is not touching the image border. Then the y-axis boundary can be determined from the initial result by loop though all the contours in that group and find the minimum y point and maximum y point. Then the image is cropped based on these boundaries Fig. 3 (d). Furthermore, from the initial result, if there are two characters that are not touching the border then the skew angle can be calculated using the same principle Fig. 3(e). Then deskew operation can be performed on the cropped image so that the accuracy of convolutional neural network in second phase will be improved. Finally, the top and bottom of the image is padded with white color. From Fig. 3(f), there are still two vertical borders that are not removed. Next step, find the connected component again using findContours function in OpenCV library Fig. 3(g). The outer most contour clearly identifies the border. The mask Fig.3 (h) is created and do a not xor operation with the image. Also, if there is a contour that is too close to the vertical border, it will get removed too.

After all these preprocessing steps, the method proposed before was performed again on the image. This time, the result looks pretty promising Fig. 3(i).

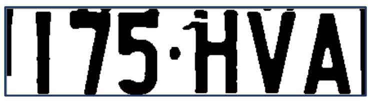


(a)

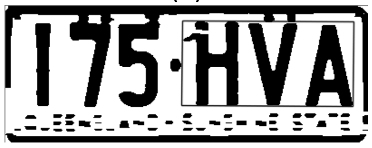

(b)



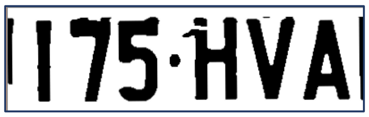
(c)



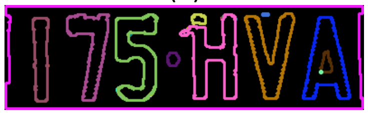
(d)

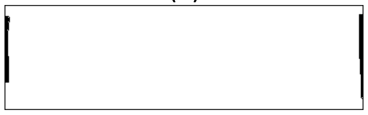


(e)



(f)


(g)


(h)

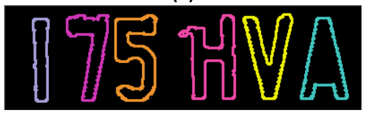
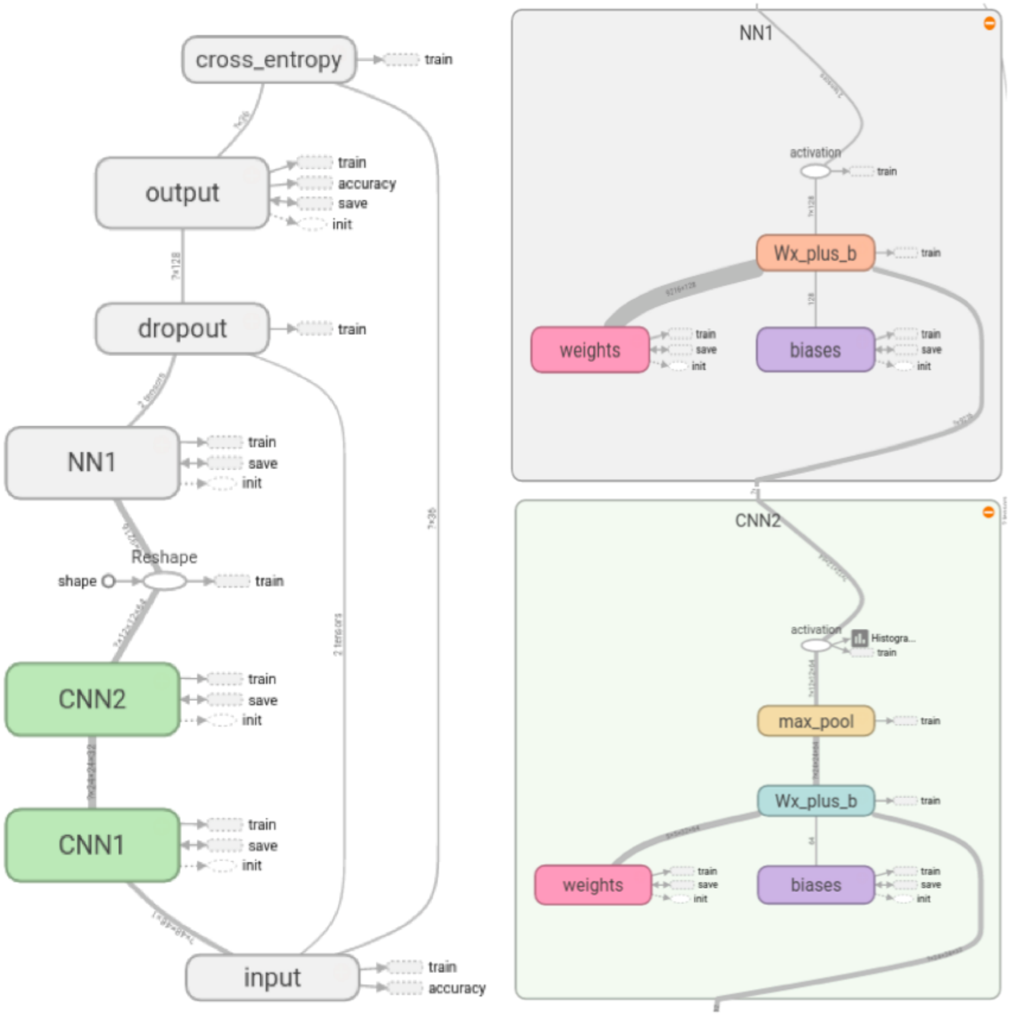

(i)

Fig.3 (a) Binary Image. (b) Contours of the binary image. (c) Result that produced by the method proposed. (d) Cropped image. (e) Rotated Rectangle. (f) Image after deskew and padding. (g) Contours of image after deskew and padding. (h). Border mask. (i) Image after removing border

The second phases of the project was implemented using Python with numpy and tensorflow library. Numpy library was used to do matrix calculation and tensorflow was used to build a convolutional neural network. Tensorflow is an open source library for machine learning originally developed by researchers and engineers of Google Brain Team. The objective of this phase is to classify the character image outputted from the first phase into 36 classes (A-Z and 0-9).

The neural network that is used in the project consists 5 layers including input layer and output layer and 3 hidden layers. The data set is characters from computer fonts with 3 variations such as italic, bold, normal. It consists 36576 instances with 1016 instances per class. Font images were downloaded from this website http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/ and divided them into 3 groups as training images, validation images, and test images. These images are labeled with 36 classes.

The structure of the convolutional neural network:



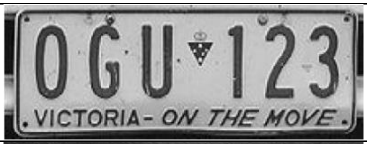
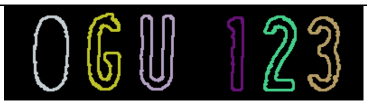
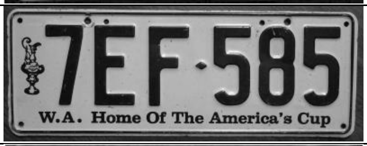
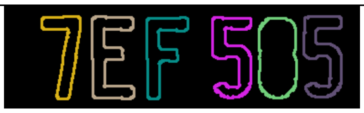
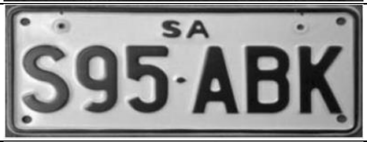
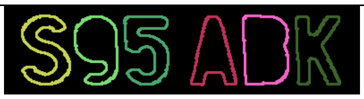
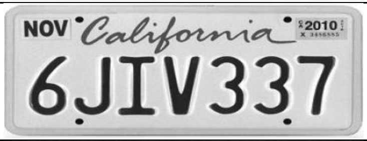
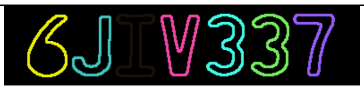
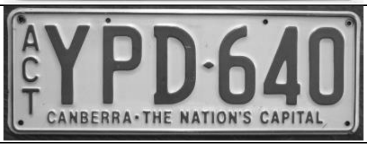
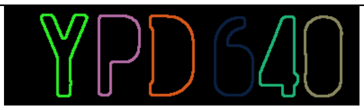
The values of each parameter are as follows. Batch size = 50, Patch size = 5, Learning rate = 0.0001.

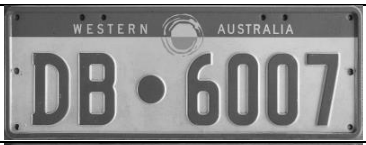
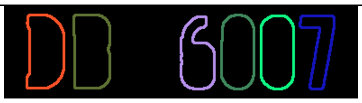
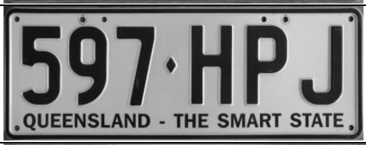
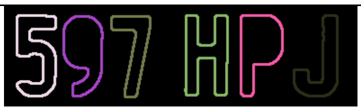
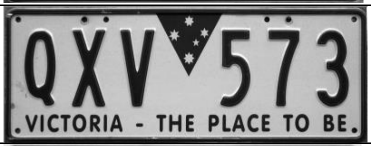
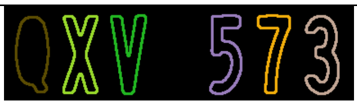
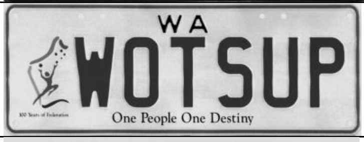
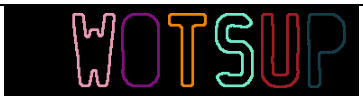
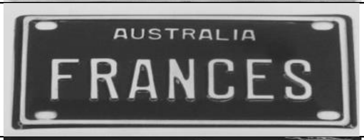
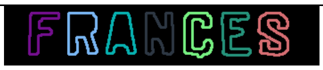
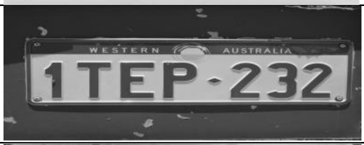
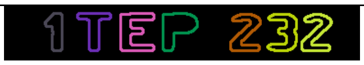
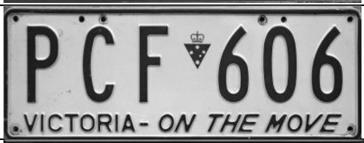
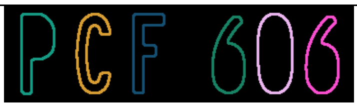
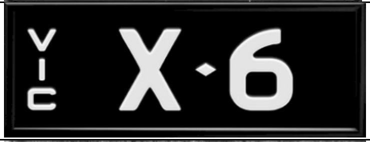
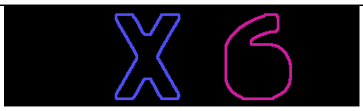
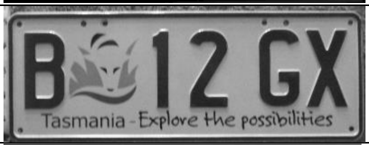
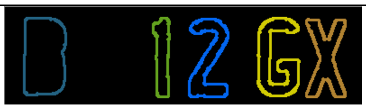
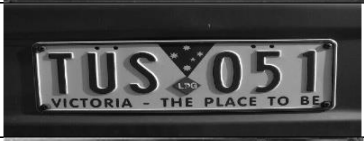
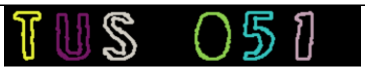
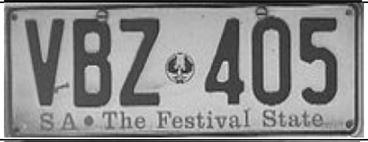
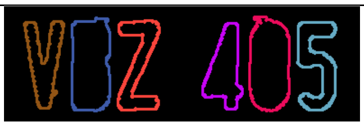
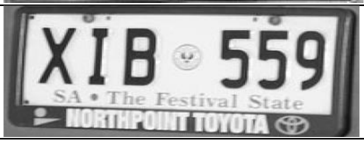
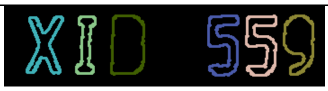
Rectified Liner Unit function (ReLU) was adopted as an activation function. The size of training images is 128 pixels by 128 pixels and it is too large for efficient computation with limited resources. Thus the size was reduced to 48 pixels by 48 pixels.

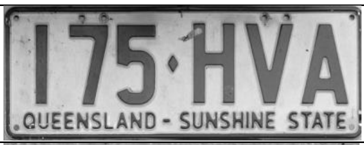
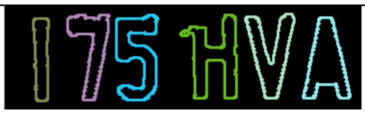
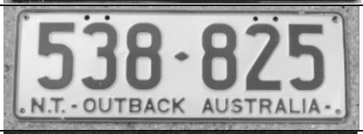
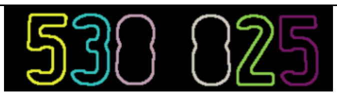
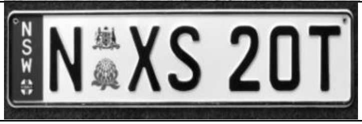
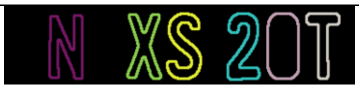
Two convolutional layers in the graph are almost identical with a max pooling layer inside it to reduce the sample size. The only difference is the depth which the first one is 32 and the second one is 64. First CNN applies $5 \times 5 \times 1$ filter to the input image and gets 32 convolution layers. ReLU function is applied to outputs of this filter and max pooling is also applied to them to reduce the dimensionality of the representation. Second CNN does almost same things, although this layer generates 64 convolutional layers instead 32. After second convolutional layer, outputs are reshaped into one-dimensional array to be passed to next fully connected layer. The third layer is a normal fully connected neural network that has 128 hidden layers. However, in order to prevent it from overfitting, the dropout layer is added. This method apply Bernoulli function to the input before calculate $\mathcal{W}x + b$ [8]. Therefore, x becomes $x \times Bernoulli(p)$ with dropout layer. When training, the probability of dropout is 50% namely x becomes $x \times Bernoulli(0.5)$. Finally, outputs of third layer are passed to output layer that has 36 hidden layers and applies soft max function to calculate probability of each class. After CNNs made prediction for training images, Adam algorithm [5] is used to minimize the cross entropy of the prediction label and actual label. This training procedure was iterated 107000 times with 50 training images each time and the final precision was about 97.8% on validation data and test data.

5. Experiment and Results:

The following are 18 license plate samples:

#	Input	Phase 1	Phase 2
1			0GU123
2			7EF585
3			S95ABK
4			6JIV337
5			YPD640

6			DB6007
7			597HPJ
8			QXV573
9			W0T5UP
10			FRANCES
11			1TEPZ32
12			PCF606
13			X6
14			B12GX
15			TUS051
16			V8Z405
17			XIB559

18			I75HVA
19			538825
20			NXS20T

From the result, Phase 1 appears to be very promising. The program has successfully segmented all the character from all the images despite the varying illumination. In phase 2, there are 118 character need to be classified, and the classifier got 3 wrong classifications. It seems to confuse about 0 and O, I and 1 and S and 5. Other than these characters, it gets everything right. The precision in phase based on the 20 images we got is 115/118 = 97.45%.

6. Conclusions

In this project, our group had a hands on experience to a modern computer version problem and we are very proud of our result. In terms of further work, the method proposed in the first phase can only detect one line of characters in license plate. Some statistical method can be included to calculate the possibility of a group being a character lines in order to detect multiple lines. Furthermore, L2 regularization can be applied to the convolutional layer in order to further prevent overfitting. The training dataset is computer generated pictures with no noises at all. In order to improve the performance of character recognition, natural images can be added. Moreover, license plate localization in an image is not required in this project. However, convolutional neural network based method can be implemented if there are enough training samples.

7. Learning outcomes

In this project, several image processing techniques was applied such as noise reduction, morphology based operation, connected component analysis. Besides, a neural network with two convolutional layers and one fully connected layer was implemented. The result is pretty promising. OpenCV in C/C++ and tensorflow in python is familiarized during the project.

8. How does one use your source code?

The project divides into two parts which the first phase was implemented in C/C++ and second phase was implemented in Python.

Phase 1:

1. C/C++ compiler
2. OpenCV library
3. Change LDFLAGS in Makefile to OpenCV lib path [Makefile is provided]
4. Make

Phase 2:

1. Python 2.7
2. Install all the dependency in requirements.txt. (pip install –r requirements) [some package may not be able to install properly, please check if the system meet all the requirements] Tensorflow only runs on Linux and Mac
3. To train the model, run "python model.py"
4. To evaluate a license plate image, run "python eval.py –f [file path]" [it will automatically run "./main [file path]" using subprocess module]

Since we cannot upload large files when we submit, we have uploaded our project to GitHub. If you want the full project with the training sample please download at:

https://www.dropbox.com/sh/3itdlgraz68audk/AADptWtginvNCwCdkyjfhto7a?dl=0

9. Reference:

[1] Shan Du, “Automatic License Plate Recognition (ALPR): A State-of-the-Art Review“, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 23, NO. 2, FEBRUARY 2013

[2] D. Zheng, Y. Zhao, and J. Wang, “An efficient method   of license plate location,” Pattern Recogn. Lett., vol. 26,   no. 15, pp. 2431–2438, 2005.

[3] C. Wolf, J-M. Jolion, “Extraction and Recognition of Artificial Text in Multimedia Documents”, Pattern Analysis and Applications, 6(4):309-326, (2003).

[4] Hui Li, Chunhua Shen, "Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs", [online] http://arxiv.org/abs/1601.05610 (2016)

[5] Kingma, Diederik P. and Ba, Jimmy, “Adam: A Method for Stochastic Optimization”, Proceedings of the 3rd International Conference on Learning Representations (ICLR), kingma2014adam, (2014)

[6] Ruder, Sebastian. "An Overview of Gradient Descent Optimization Algorithms." Sebastian Ruder. 2016. Accessed May 24, 2016. http://sebastianruder.com/optimizing-gradient-descent/index.html#gradientdescentoptimizationalgorithms.

[7] N. Otsu, “A threshold selection method from grey level histogram”, IEEE Trans. Syst. Man Cybern., vol. 9 no. 1, pp. 62-66 (1979).

[8] Nitish Srivastava, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.”, Journal of Machine Learning Research 15 1929-1958, (2014)