# CSC420

# Project report


# Zhengming Zhou

# 1002400724

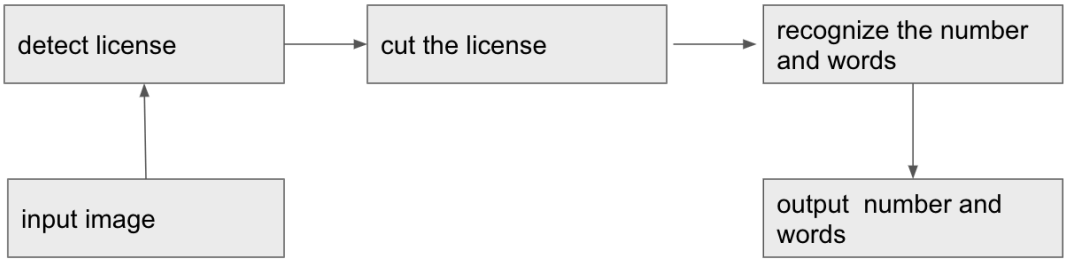

# Car License Detect and Recognize

# Abstract

This is a project that aims at recognizing the license plate of the cars. This project would use Neural Network to recognize the numbers and words, which is not often used in industry, and it may improve the correct rate that the machine recognizing the license plates.

# Introduction and Related Work

This project is about license detecting and recognizing. The reason why I choose this topic is that it is a continuous process that combine the computer vision with machine learning. This would be an interesting process since there would be many interesting connections between each of the steps.




The methods involved in this process includes morphological transform, canny edge detect, image cutting and neural network.

# Methods

There are 3 main steps involved in this process. First, I detect the license in the picture with the method that could select the rectangle in the picture. Then we cut the plate into single numbers with the method that processing binary images. Finally, we put each of the number or letters into a neural network and train a model with EMINST to use the model to recognize the numbers, so that the number would be recognized.

The specific of these 3 steps.

1. **Detect the license**

a. Use OTSU to binary the image.

OTSU algorithm is to find a threshold value which could minimize the weighted within-class variance given by the following equation.

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

b. Use opening and closing operation to denoise.

```
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
```

Result:

```
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

Result:

c. Use canny edge detect to find the edges.

Canny edge detect is the process that use the gradient to detect the edge in the picture

d. Use open and close operation again to get the potential area.

e. Find all rectangles and select them by color

f. Return the image

**2. Cut the license**

a. First use Gaussian Blur to denoise.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

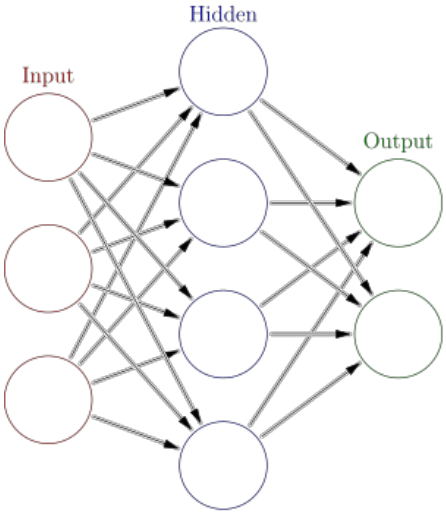
Gaussian is the process that smooth the image, and denoise the noise points in the pictures.

b. Then use OTSU to binary the image.

c. Then loop over the image and if there is a column that 93% of pixels are white, then stop.

d. Then Save the image.

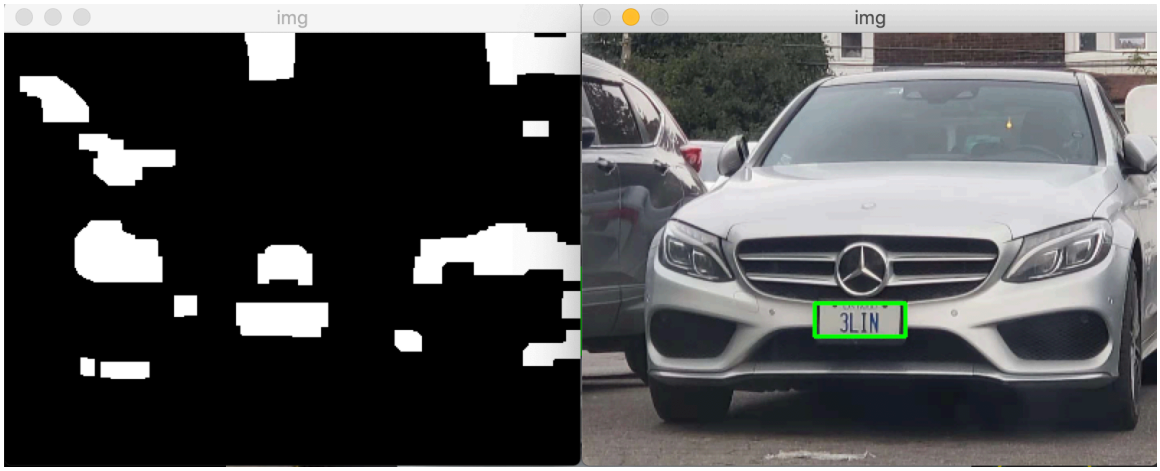
3. Neural network to distinguish the number and words.




Since the number is made into binary pictures, so it could be recognized by model which is trained by EMNIST dataset. This dataset includes handwritten words and numbers. Since the hand-written words are very similar to words in license plate, so the model could predict the input images with the models. I used the CNN to train a model that takes the input of image of shape 28*28 and give the result of recognizing with class label, which indicates what is the result of this image.

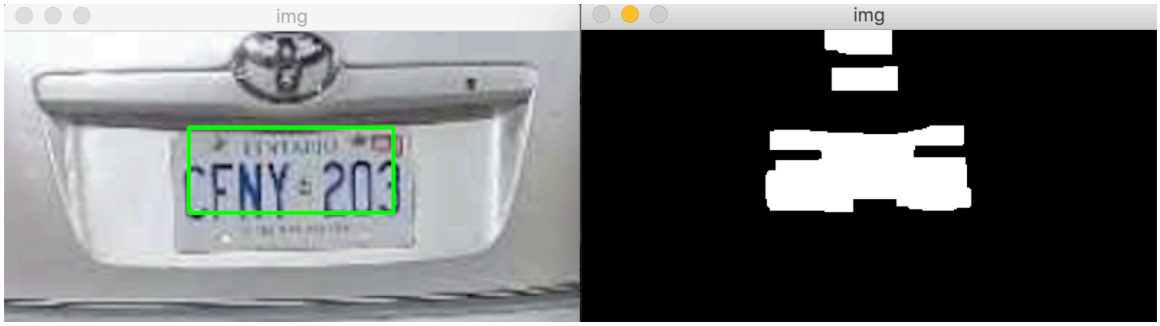
# Results and discussions

**Result of Detect the license.**




This image tends to be a successful result since the license plate locates in the middls of the car, strainght in the picture. In the precess of finding a good shaped

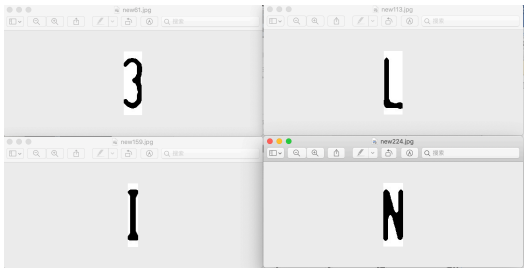
rectangle, the license plate area would be easily be selected.




This tends to be a result that is not so good. Since the area selection method is based on the blue area, this picture has quite a large part with blue, and the quality of the image is not very good, so in the process of finding a rectangle, the location is not selected accurately.

**Result of cutting the license:**


Threshold = 0.9



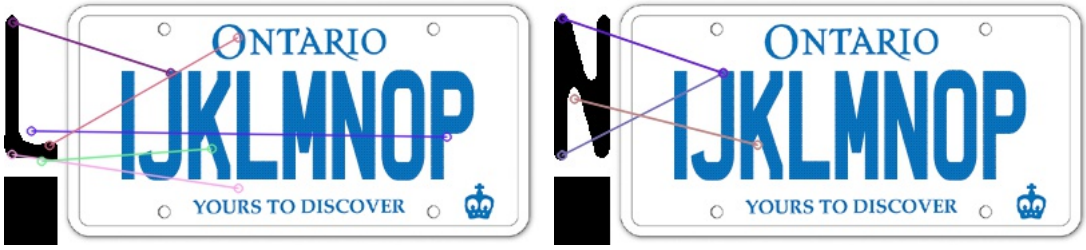

Threshold = 0.93

Different threshold could result in different results, for example, if we set the proportion of white to be 0.1, L may not be correctly cut. However, if we set the threshold to 0.93, L would be correctly cut.

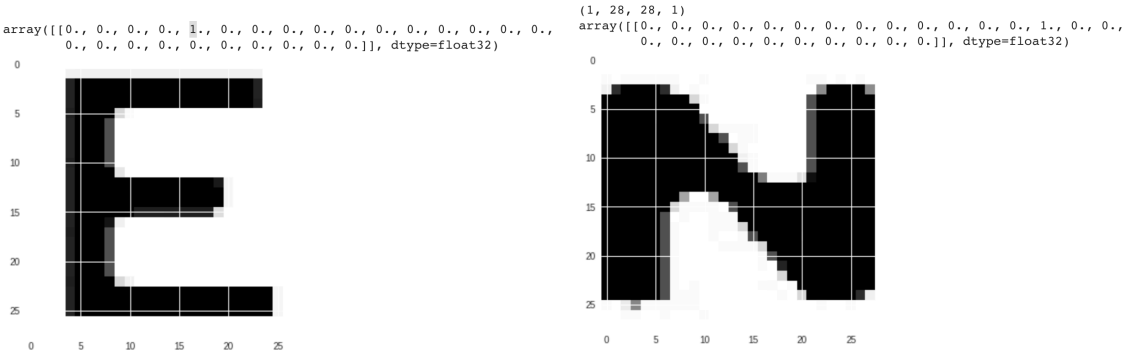
**Result of recognizing words.**

There are 2 ways that I planned to recognize the words.

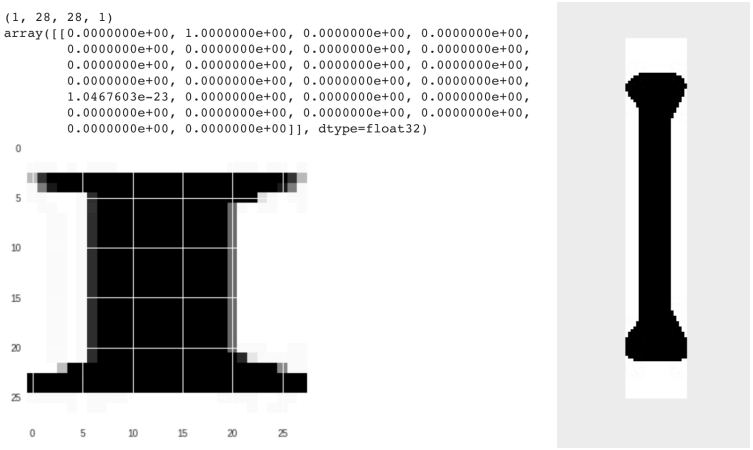
First way is to use feature matching, however, the binary image has very few information in it, to the result turns to be unsuccessful.




Second way is to use a neural network to train a model to recognize the words.




Some examples that extracted from previous steps, the model could successfully recognize words and give correct corresponding labels.




However, the word “I” is not recognized. This is mainly caused by resizing process. Since the original image is a long slim image, but the input of model is 28 * 28 image, after reshaping into proper input, the image become a shorter and the middle part is not so slim as before, which makes many noise points in the input, so the output is not correct.

# Main Challenges

There are some challenges in my project.

First tune the parameters in the models and cutting the words. In order to solve the

parameters tuning in neural networks, I look into the website, and see some parameters that often used. Then I try to make some modifications that gradually comes to the correct answers. In order to solve the parameter tuning in cutting the words, I try to print the column of the binary pictures, to see if a column is really needed, what is the actual proportion of the black pixels, and set the parameters to this.

## Future work and confusion

What I did.
Implement a project that could distinguish the license plate. It is implemented by distinguishing the license plate, cutting the license plate, and lastly use EMNIST to train a model that could distinguish words and letters.

Future Work
This project sometimes does not work when the photo is not horizontal or there are too many noises in the picture. The future work includes auto rotate the picture to make the number in the picture straight in the middle, and denoise the picture so that the work could fit more cases.

## References:

EMNIST Data set
Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. Retrieved from http://arxiv.org/abs/1702.05373

OTSU
https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html

Morphological Transformations
https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

Canny edge detection
https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html