# REAL TIME LICENSE PLATE DETECTION BASED ON MACHINE LEARNING

by

Yu-Liang Chiang

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Engineering

at

The University of Wisconsin-Milwaukee

May 2021

# ABSTRACT

REAL TIME LICENSE PLATE DETECTION BASED ON MACHINE LEARNING

by

Yu-Liang Chiang

The University of Wisconsin-Milwaukee,2021
Under the Supervision of Professor Yi Hu

License Plate Recognition (LPR) is a very useful technology skill for our world, and also includes a wide range of applications. During my student life in the US, it is not common to see this kind of technology applied in the public parking lot. LPR has been researched for many years in Asia. There are a lot of LPR systems, which include different types of technologies, being used in Taiwan. In my point of view, those systems are still using the conventional technology, which does not involve artificial intelligence. In many applications, that makes it possible to make mistakes in using those systems. In this thesis, we present a hybrid system which not only located the license plate but also recognized both the letters and the digit numbers on the plate.

In this thesis, we present a license plate system which includes recognition and location. The problem of miss-location could be solved by using an artificial intelligence system. After the miss-location is resolved, the accuracy of the text recognition will be increased. Both location and recognition systems would be trained by using artificial intelligence. The system of location would be trained by using the YOLOv5 network, and the recognition would be implemented by using the Tesseract-OCR system. It is the first time that the YOLOv5 is used on the license plate tracking.

To

my parents,

my family,

and my classmates from CYCU

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# I. Introduction

In Asia, many big cities include different types of systems to assist the city's rotation. Those systems, which always work without human control, not only decrease the expenditure but also make citizen's lives more convenient. For example, the License Plate Recognition (LPR) system is the most famous one which is settled up in many public parking lots frequently, and it helps people pay money more efficiently. In the United States, most of the public parking lots are still running the traditional system. Those traditional systems make customers take the kind of ticket to get in, and even distinguish the different cars by using that ticket. It is inconvenient that the driver should take the ticket while driving the car. If customers don't want to take and keep the ticket all day long, the LPR system will probably fix those problems.

The license plate recognition system is always separated into two parts. In the first part, the system performs license plate location by using the color and shape. The method always uses HOG histogram and HSV (hue, saturation, value) color space for feature extraction. After the license plate is extracted, it would use the SVM (Support Vector Machine) for classification. The above methods all require manual intervention. In the second part, character segmentation also includes similar techniques to cut and recognize the text. Although the traditional license plate recognition probably has great performance during some situations, it still needs to consider the varied conditions. We need to implement many different situations to adjust the system during the experiment. The system always performs miss-location when the light gets dark or during the weather change. If the system doesn't cut a license plate successfully, the recognition part will be uncompleted, as shown in Fig.1.

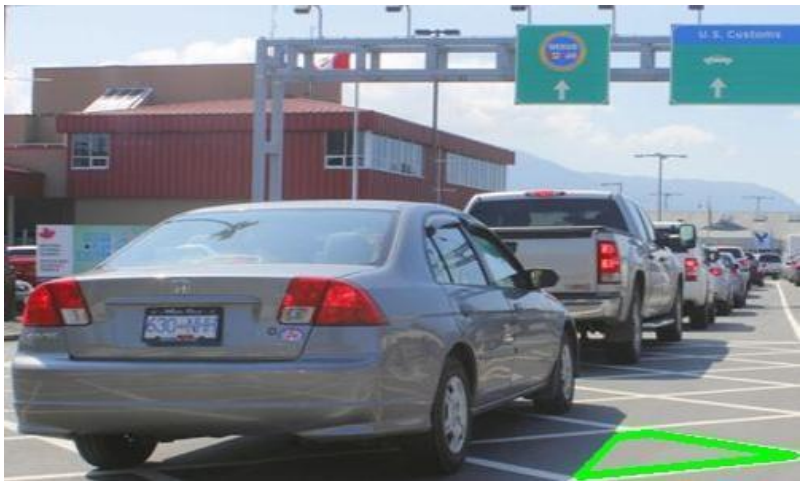

Figure 1. Miss-detection demonstration

Because the miss-location problems are noticed, the license plate detection system, which is based on machine learning, grows very fast. Currently, the image recognition research are mainly divided into three categories: “Image Classification”, “Object Detection”, and “Instance Segmentation” which are shown in Fig 2.

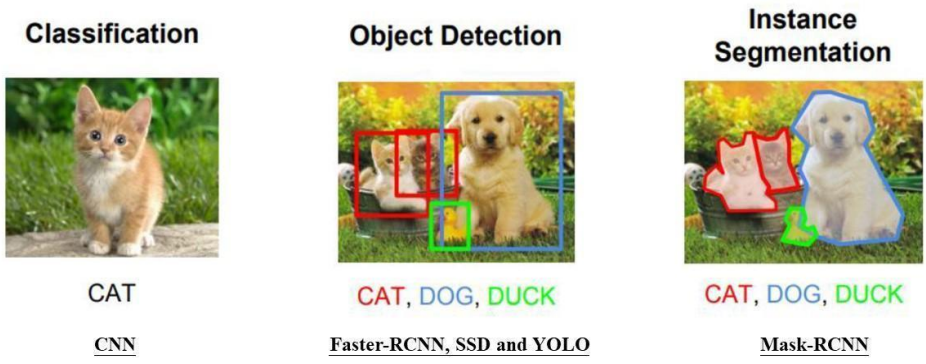


Figure 2. Three different types of the neural network [2]

All the models of the detection system are based on convolutional neural network (CNN) and have been widely used for LPR [1]. First, the CNN method is more like the image classification. Second, typical representatives are the R-CNN (Regions with CNN features) which include Faster-RCNN and Mask-RCNN, the YOLO (You Only Look Once), and the SSD (Single Shot MultiBox Detector) [1]. Both Faster-RCNN and YOLO belong to the object detection part. Mask-RCNN is present by instance segmentation [2]. The object detection, which fits the normal application, is widely used in the LPR research. Although many different types of object detection applications have appeared, not all of them could be used in real time situations. For example, Faster-RCNN and Mask-RCNN are two neural networks which are impossible to implement on the simple device with weak performers, since the heavy network.

In the LPR system, digit numbers and letter recognition always played an important role. There are several techniques, which can identify text, used in many applications. Not only traditional image processing but also deep learning has great performance in the recognition area. It will cost too much time to prepare the previous work in both conventional technologies and deep learning technologies. There is an easy way, which doesn't need to train the model or implement heavy work, called "Tesseract-OCR(Optical Character Recognition)". In a nutshell, Tesseract is all about making information available to users, and when this information is in a paper document, OCR is the process by which we can convert the pages of this document into text that can then be used for indexing [3]. It has Unicode (UTF-8) support, and it can recognize more than 100 languages "out of the box". It can be trained to recognize other languages and supports various output formats: plain-text, hocr(HTML), pdf [4] [5].

In this thesis, the goal is to design a system which is implemented on the camera, not only detect the license plate but also recognize both letters and numbers. This approach consists of license plate location and character recognition. Because of the high accuracy location required, we select to use YOLOV5 which doesn't include a heavy network but with strong performance. We also implement the Tesseract-OCR system in the character recognition part for saving the experiment time.

In this thesis, we make the following contribution:

- It is the first time that YOLOv5 is used in license plate recognition. YOLO is a kind of neural network without a heavy network so that it can be operated on a small device.

- The license plate recognition systems, which included Tesseract-OCR, used to be implemented with the conventional technology which doesn't include machine learning. It is also the first time that Tesseract-OCR is combined with the machine learning method.

The remainder of this paper is organized as follows. In Section II, we introduce the relevant theoretical basis that we already used in this thesis. Soon after, the new system, which we presented, would be described in Section III. The research results, comparing data, and testing images are shown in Section IV. Finally, the conclusion and future research directions are shown in Section V.

# II. Related Work

In this section, we would introduce some related works which are already implemented on the license plate recognition. The human eyes and brain can judge and recognize objects at the same time. The detection and recognition are separated into two parts during the system. The computer will discover the areas which may contain the target and then classify those areas into predefined categories. Deep learning has significantly improved computer detection capabilities. Deep learning can be roughly divided into supervised and unsupervised. The current research on license plate recognition is still based on supervised learning.

## A. Convolutional Neural Network (CNN) [6]

CNN method is the core of many recognition and detection technologies, and it has been widely used in various fields in recent years. The basic architecture of a CNN is combined with the convolution layers and the pooling layers, and it adopts multiple convolutional layers and pooling layers to learn the characteristics of the image content from each pixel value of the original image. CNN usually uses at least more than three layers. Therefore, CNN is classified as deep learning, and it usually includes the following layers which are shown in Fig 3:

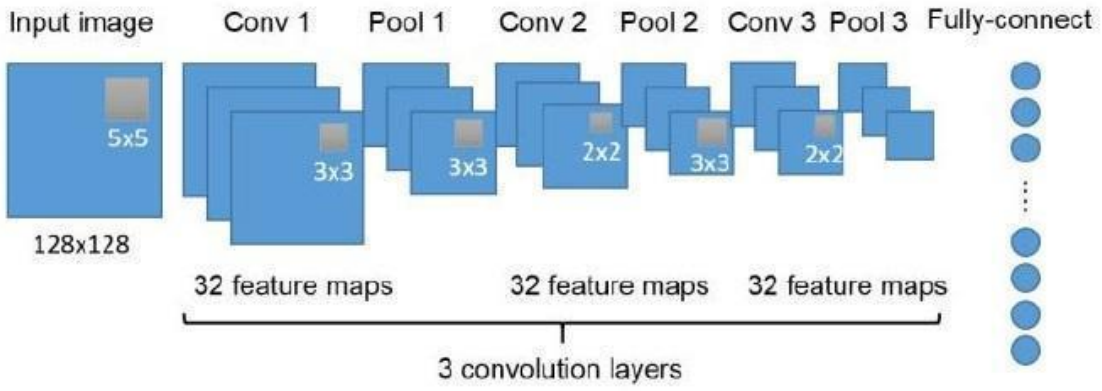


Figure 3. A basic CNN structure [7]

a) Input layer

The input layer is related to the data related to the input image. It usually contains the basic information of the input image, such as the size, length, width, and quantity of the image.

b) Convolutional layer

The convolutional layer is convolved from the previous layer, and then stores the parameters and weights of the training results. The output of this layer is usually smaller than the length and width of the input layer but includes deeper depth. This layer stores the training weights as computable parameters. The parameters of $k_1$ and $k_2$ are the kernel size. Both $N_{input}$ and $N_{output}$ are the amount of the input and output filter, and them $N_{bias}$ always has the same amount comparing to the $N_{output}$:

$$N_{param} = k_1 \times k_2 \times N_{input} \times N_{output} + N_{bias} \quad (1)$$

c) Activation Layer

This layer is used to activate the convolutional layer. The activation function is usually linear because it only changes the input value of a negative

number to zero while retaining the input of a positive number. This is the rectified linear unit function (Rectified Linear Unit, ReLU), also known as the modified linear unit.

d) Pooling Layer

This layer is to subsample the previous layer and search for the best features of the input image and use the kernel size to define the output size of the input tensor. For example, if the kernel size is 2, the output size is divided by 2.

e) Fully-connected layer

The fully connected layer is the last layer that stores the weight of the training result during the training and is also used as a classifier. Usually, the convolutional network will straighten the cuboid obtained at the end into a one-dimensional vector and send it to the fully connected layer to cooperate with the output layer for classification. The number of weights and parameters stored in this layer:

$$N_{param} = N_{input} \times N_{output} + N_{bias} \quad (2)$$

## B. Faster-RCNN

In R-CNN, multiple local regions must be calculated separately, and many areas overlap with each other. In other words, many convolutional neural networks are repeated operations. Fast R-CNN only calculates the convolutional neural network once, and the features, which are extracted by the convolutional neural network, can be shared with all candidate regions [8].

Faster R-CNN is an improved version of Fast R-CNN. The backbone of Faster R-CNN is ResNet, and with RPN to find the predicted bounding boxes [9]. Faster R-CNN, which abandoned

the previous regional proposal and selects regional proposals from the CNN function map, has improved the implemented time.

### C. Mask-RCNN

Mask R-CNN is a two-stage architecture. The first stage generates candidate regions, and the second stage classifies the candidate regions and generates both borders and masks. Mask R-CNN is built on Faster R-CNN. The difference between Mask R-CNN and Faster R-CNN is that it replaces RoIPool, which performs feature extraction, and uses RoIAlign to solve the problem of correctness [8].


Figure 4. LPR system based on Mask-RCNN [9]

Although those RCNN methods implement with high accuracy, all of them need a heavy computation workload to run smoothly. It is difficult to set up these RCNN methods on the low computation device, especially for cost reasons. That is the reason why we select the YOLO network for our research.

# III. Technical Approach

## A. System Pipelined

When implementing the machine learning application, it is always separated into three major parts: training data collection, training model, and prediction of the object by training model. As shown in Fig 5, we would collect the training data first. After the amount of training data reaches our goal, those training data would be sent into the YOLO network to train the model for predicting. When the training model is finished, the users would send their testing images to the training model for the image recognition. Finally, the results will show to the user on the interface.

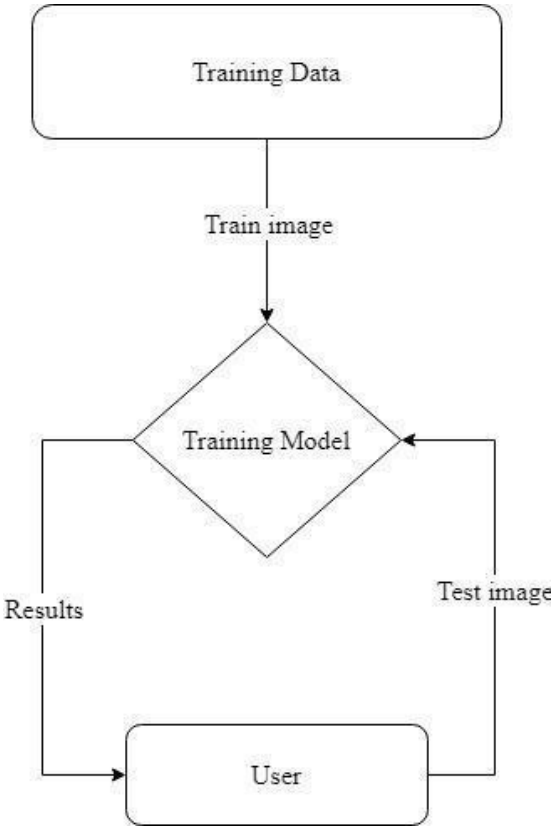


Figure 5. Briefly architecture of Machine Learning

As shown in Fig 6, it is our system architecture. In our system, it includes two major parts. The first part is that we will detect where the license plate is. It is a kind of location. After our system located the license plate correctly, it will cut the part which includes the license plate, and then send it to the next step. In the second part, we use the "Tesseract-OCR" system to identify both digits and letters. We also will add some image processing techniques to improve the Tesseract-OCR's performance.

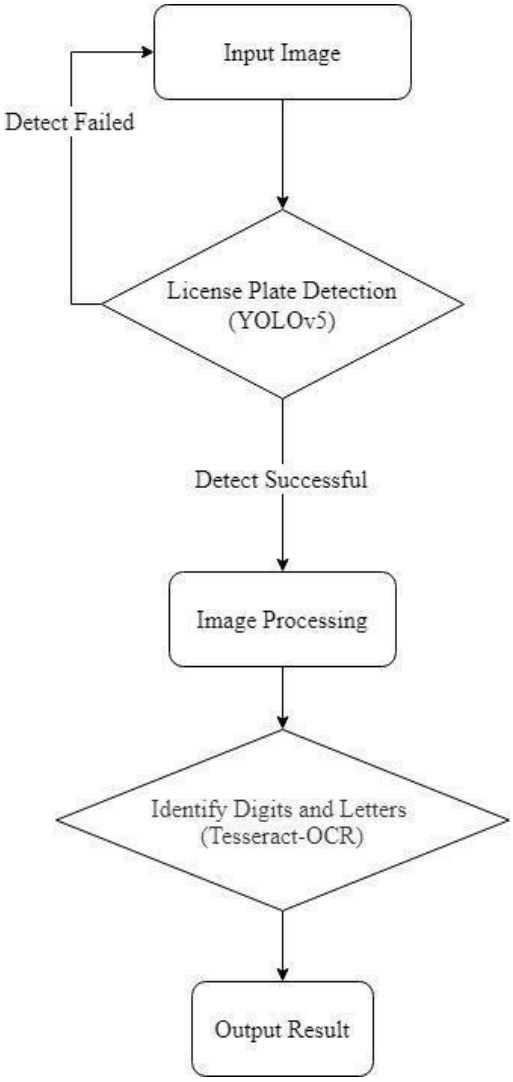


Figure 6. System Architecture

## B. Data Collection

Data collection always plays an important role during training the model. This major problem has a big connection with the performance of machine learning. Although there are a lot of free databases, it is still not enough for us to reach the highest accuracy. Therefore, collecting enough data would be the first problem that we need to solve. In our research, the license plate is the main target that we need to collect. In the United States, there are different colors of license plates from different states. Because of this unique situation, we only select license plates, which belong in Wisconsin, to be our training data. Thus, how to collect the data efficiently will be my first problem. During this experiment, we need almost thousands of images to be the training data. It is hard to take a photo one by one, so we figured out some ways to collect the data.

a) We try to set up a smartphone beside the road to take a video. In this study, MATLAB was used to extract images from the video. As we know that there are 60 or 30 frames in a 1-second video, we try to extract every 20 or 10 frames for network training to avoid duplication.

b) After extracting the image from the video, the amount of data is still not enough. We try to rotate the image. Rotating the image is a very common way to increase the amount of data, especially during the model training. The images are turned 90 and 180 degrees for each by us. Thus, we received three times a big amount of data. There is a part of our data which are shown in Fig 7.

c) The third step is called labeling. In this part, we used the tool which is already applied to the python environment called ‘LabelImg’ to label the image. By using this tool, we should figure out the boundary where the license plate will be, and then store the location into an XML file, as shown in Fig 8.

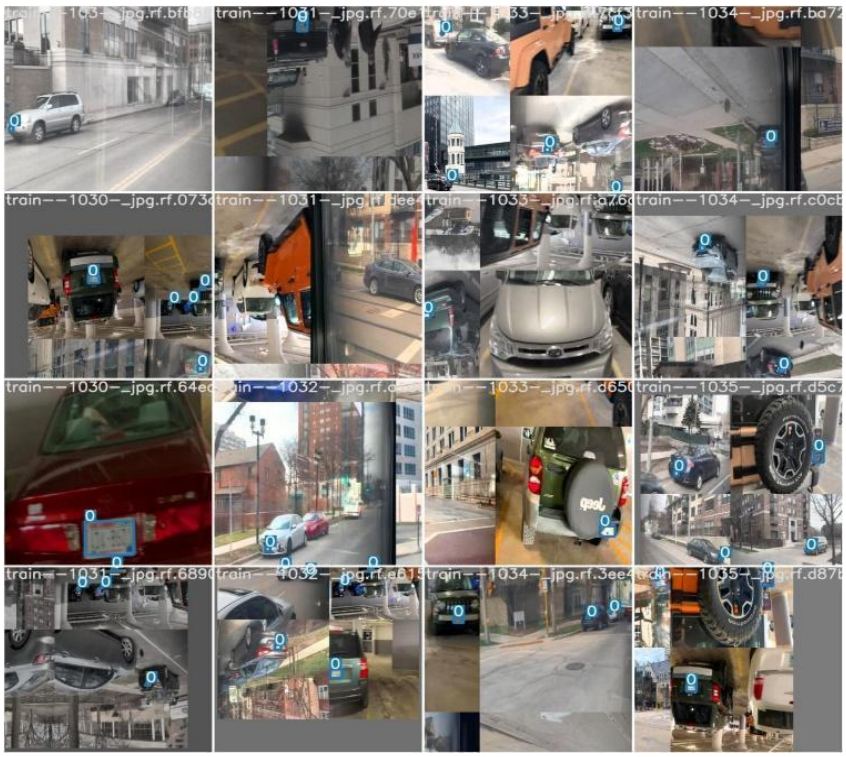

Figure 7. License plate data.

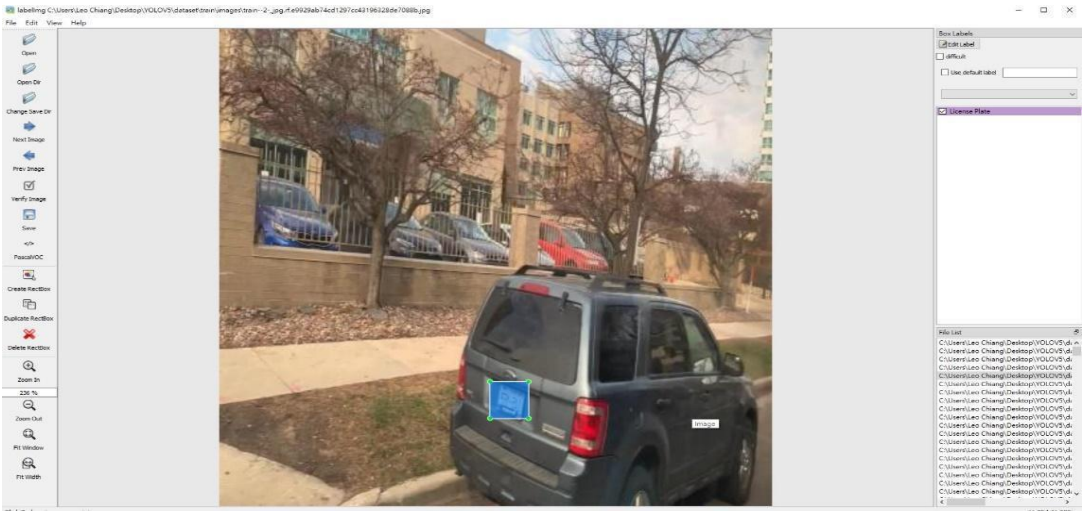

Figure 8. Labeling images using the ‘LabelImg’ tool

## C. License plate detection

In the part, we try to use the data, which is already labeled, to train the YOLO version 5 network. As we know, it is impossible to build a super-powerful computer in the public parking lot. We select YOLO, which is without a heavy network, to train the license plate model. Before testing the image, the YOLO detection system will resize the testing to $448 \times 448$. After resizing the image, it will run a single convolutional network on the testing image, and then thresholds the resulting detections by the training model [11], as shown in Fig 9. The Architecture of YOLO has 24 convolutional layers followed by 2 fully connected layers. Alternating $1 \times 1$ convolutional layers reduce the features space from preceding layers [11].

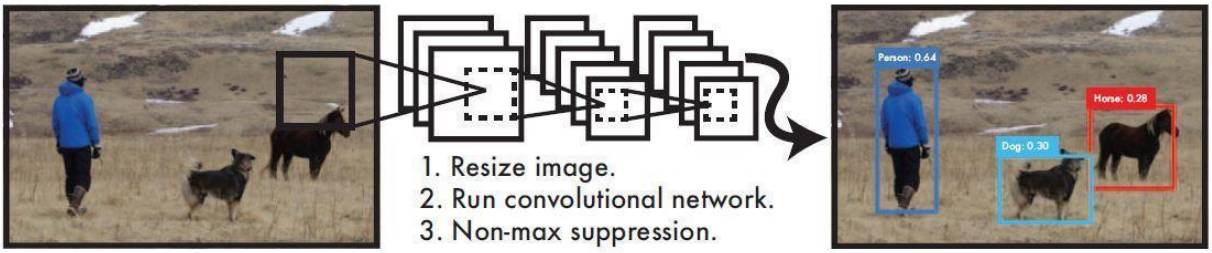


Figure 9. The YOLO Detection System [11]

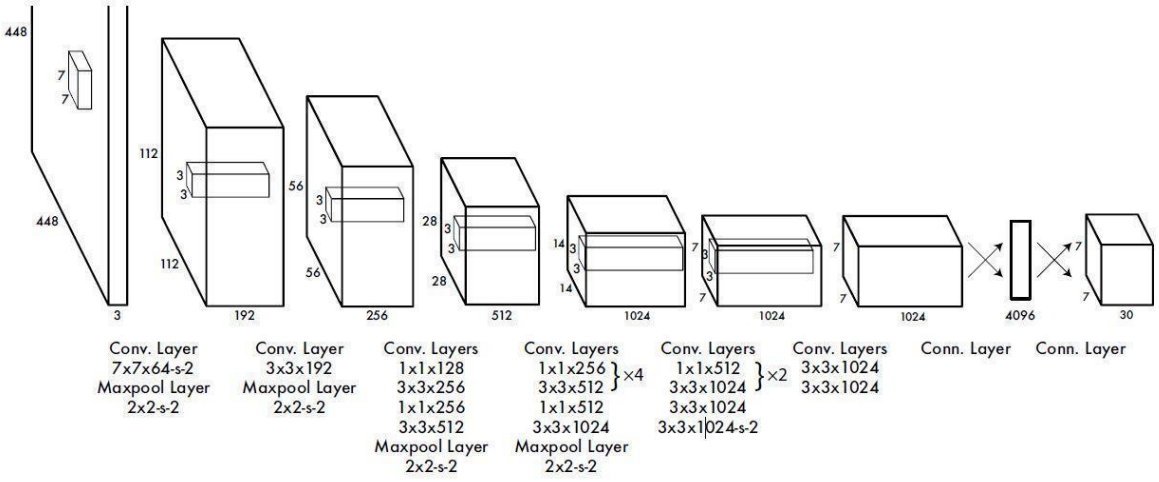


Figure 10. YOLO network structure [11]

This network's final layer predicts both class probabilities and bounding box coordinates. It is very similar compared with the object detection system which is based on RCNN. As the other network does, the YOLO network will normalize the bounding box width and height by the image width and height so that it could fall between 0 and 1. It will also parametrize the bounding box *x* and *y* coordinates to be offsets of a particular grid cell location so that information is also bounded between 0 and 1 [11]. There is a linear activation function that can rule the final layer and all other layers.

$$\emptyset(x) = \{x, \quad if\ x > 0\ 0.1x, \quad otherwise \tag{3}$$

After the YOLO network finishes the training model, it will use the testing image to test by itself. We can calculate the early accuracy of this model by YOLO's output, as shown in Fig 11. The early accuracy can help us to analyze that this model would have great performance, and then decide that it is necessary or not to retrain the model again.


Figure 11. Testing image with the boundary box

## D. Digit and Letter Recognition

In our system, we will crop the image from the detected area with a boundary box. But the images, which are cropped from the original data, always have a lot of noise. Those noises play an important role in digit and letter recognition. Because we need to reduce the noise from the cropped image, we add some previous works before recognition, as we have shown in Fig 13.

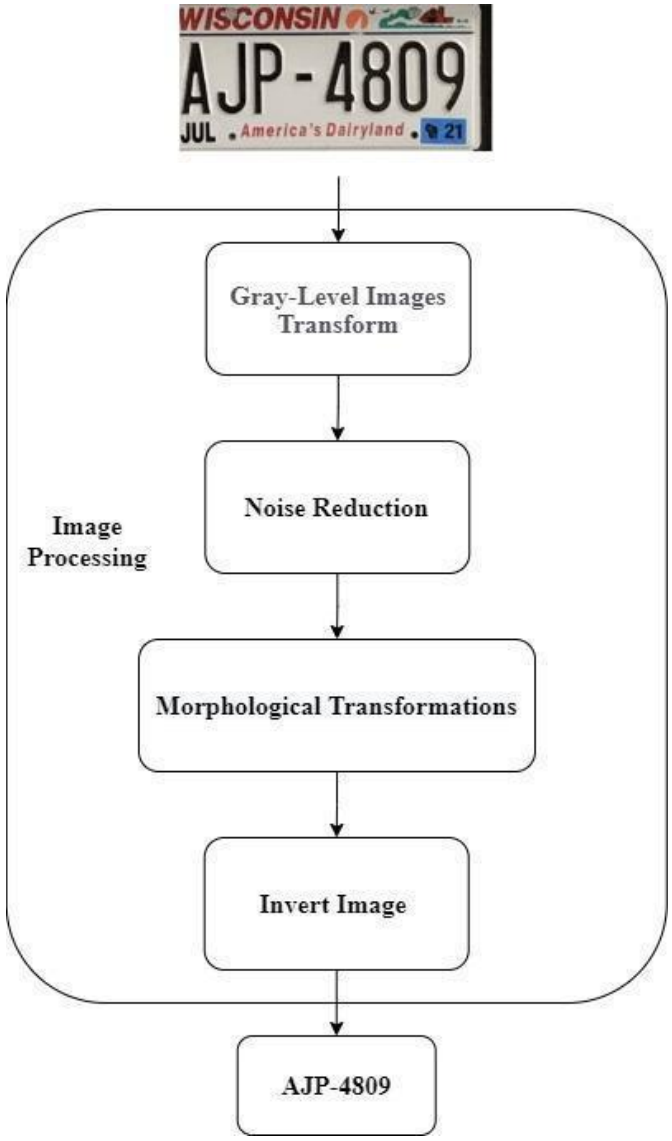


Figure 12. Image processing of our system

a) Gray-level images transform: In many image processing cases, it is a normal skill that people will transform the image to gray-level before doing the next step. The original image combines with RGB three-dimension, and it is complicated for the testing part.

b) Noise reduction: As we have known, the Gaussian filter has been used to remove the noise. The blurring effect could be controlled by how big the kernel size that we want to use. In our case, since there are lots of reflections or dust from the license plate surface, to remove that noise we would use a 3 by 3 Gaussian kernel.

c) Morphological transformations: After the image is finished gray-level images transform and noise reduced, the color of the image will be left only black and white. If we want our testing data more clearly, we need to dilate the area where the digit and letter are located, and then erosion it again immediately.

d) Invert image (color): Although the image which we want to identify is only with black and white color, we still get some error recognition during the test. This skill is to invert the color in the image, and we make the color of digit and letter location transfer from white to black.


Figure 13. Image processing before recognition

In this thesis, Tesseract-OCR is used as the final step for digit and letter recognition after the image has been sufficiently processed. We need to send our testing image into the Tesseract-OCR system. After the digits and letters on the license plate have been identified successfully, the prediction of the result will appear on the license plate, as shown in Fig 14.


Figure 14. Demonstration of our system

# IV. Experimental Setup

In this part, we provide the information of our experimental setup including computational resources, how those resources perform, datasets used during the research, training configuration, and the evaluation criteria to judge our system performance by comparing with other methods which are already used.

## A. Research setup

In this thesis, we separate license plate recognition into two major parts. It includes license plate location and text recognition. In the software part, our system is implemented in “Python 3.8”, the “PyCharm Community 2020” editor and builds the model by importing Keras 2.3.1 package to use TensorFlow 2.0 backend. In the hardware part, the computer, which is used to train the model and test the data, includes a 3.6GHz Intel Core i7-8700 CPU and NVIDIA GeForce GTX 2080 GPU.

## B. Data

As we said in the previous section, if we expect to receive the training model which is with high performance, the quality of the database also plays the most important role during training the machine learning. Although we already received a very large database which is taken by ourselves, the training model still has a high probability to crash during the training. Another key point of the training model is the percentage of training and testing part which is set by the researcher. If the training set is too much bigger than the testing set, the training model will be trained in very low efficiency with poor performers. In other words, it is important to control the amount of both training and testing set. Some of the research, which wants to increase the accuracy during the testing part, would use the same database in the training and testing set. In this worst example, the model has high

accuracy only during the research but not in the real world. During our experiment, the training set, validation set, and test set are independent of each other and are not reused. This setup could let out research be closer to the real situation and make the results of this study more valuable. The database would be divided into 80 % of the training set and 20 % of the testing set during training the model.

## C. Training model

A lot of parameters that we will probably adopt during the training, may affect the performance of the training model, such as the optimization, the number of epochs, batch size, and learning rate. In our research, we did not explore the best configuration because of limited time. We only adopt the batch size, epoch, and learning rate during the training.

- **Learning rate:** The learning rate controls how fast the model is adjusted to the model. If the learning rate is too large, it would not find the optimal solution. On the other hand, we don't want to make the learning rate too small, because then we might never end up with the right values for our weights [12]. Furthermore, it requires more epochs for updated training.
- **Batch size:** Batch size is one of the important parameters in machine learning. In general, if the batch size is too large for training, memory bursts and local optimization may occur. The Small bath size introduces greater randomness, and it is difficult to achieve convergence [13]. Normally, we would set the size in the power of 2 to fit the memory requirement, such as 32, 64, 128, and so on. According to our experiment result, the size of the batch would be set to 16 which included the best performance.

- **Epoch:** Epoch is a parameter used to define how many times the learning algorithm will work through the whole training dataset. In other words, it is a term used and indicates the number of passes of the entire machine learning algorithm has completed [14]. The number of epochs should be an integer. The size of the epoch would significantly affect the training time. Because we only have one category in both the training set and testing set, epoch would be set up into 20.
- **YOLOv5 model:** In our research, we used YOLOv5 to be our training network. There are four YOLO models which can let users select based on a different experiment. Those models are called YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, as we have shown in Fig 15 and Fig 16. According to Fig 15, YOLOv5s, which has the smallest network, takes the least time to train the model. On the other hand, YOLOv5m, which has the largest network, takes the most time to train the model. In this thesis, the system we built is to implement on the computer which doesn't have powerful efficacy, so that we choose YOLOv5s to be our training model.

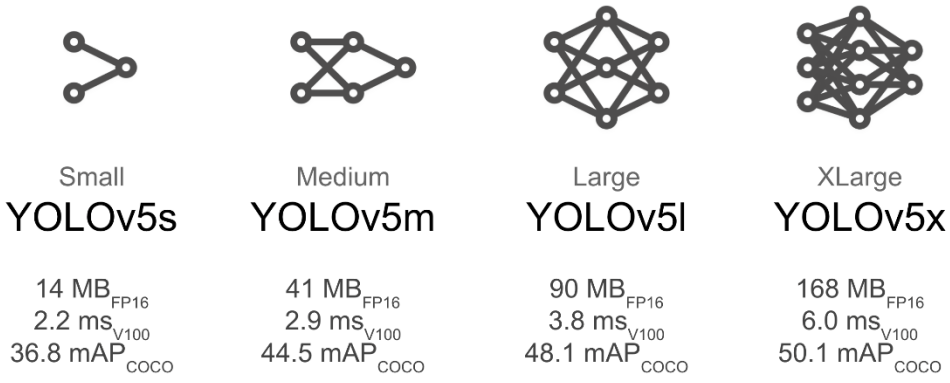


Figure 15. Four models of YOLOv5 [15]

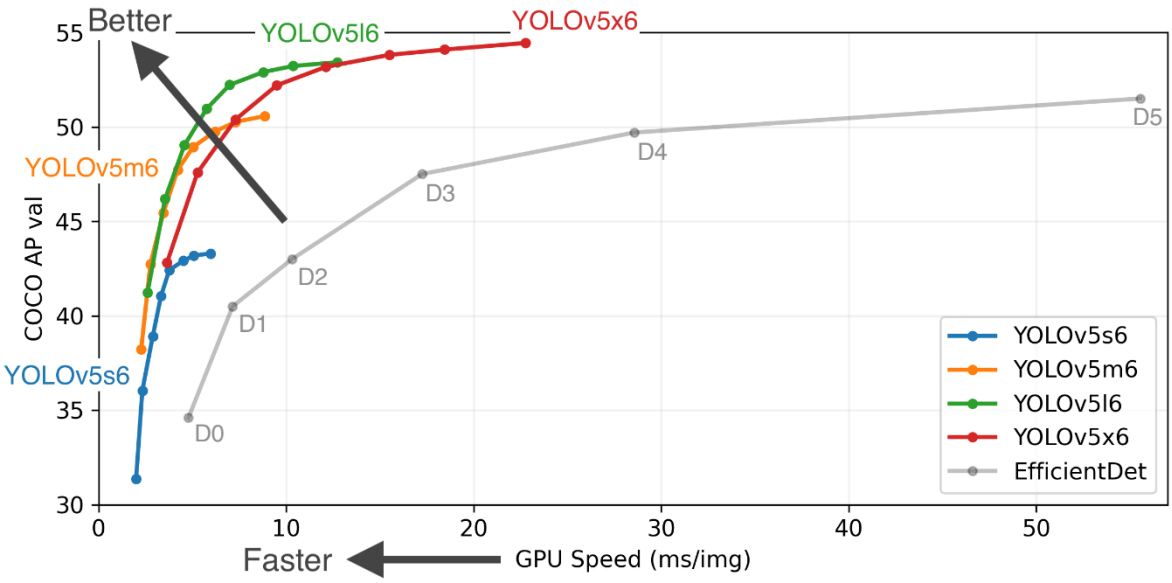


Figure 16. Model comparison [15]

## D. Performance Analysis

In this section, we try to analyze and conclude our performers of training models both in license plate location and recognition. And we also perform the accuracy of each class after the classification to demonstrate the dataset.

There are several options which let us use. Based on our system, we select a method which is called “Confusion Matrix”. In most image recognition systems, the confusion matrix can be used to evaluate the performance of one recognition system. Each column of the matrix represents the prediction or recognition result, and each row represents the ground truth of a category. The confusion matrix is named this way because it is convenient for people to judge whether a computer or machine has confused multiple objects of different categories, as shown in Table 1.

| Ground truth / Prediction | A | B |
|---|---|---|
| A | **89** | **11** |
| B | **25** | **75** |

Table 1. The example of the confusion matrix

To evaluate the recognition ability of deep learning neural networks, we called the situation, which detects the license plate successfully, as positive, and the situation, which fails to detect the license plate, as negative. The system can also interpret two situations as detect successfully and fail to detect, as we have shown in Table 2.

| Ground truth / Prediction | License Plate | No License Plate |
|---|---|---|
| License Plate | True positive (*TP*) | False positive (*FP*) |
| No License Plate | False negative (*FN*) | True negative (*TN*) |

Table 2. Confusion matrix for the license plate

The classification accuracy could also be calculated by the confusion matrix as follows. The accuracy is the percentage of correct recognition results, as in formula (3):

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3}$$

Precision is the percentage of correct license plate location to failed license plate location, as in formula (4):

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

Detection rate is the percentage of correct license plate location to images without license plates, such as (5):

$$Detection\ rate = \frac{TP}{TP + FN} \tag{5}$$

Although we already have enough databases, the performance of the training model still would be influenced by the percentage of training and data set. Because of this problem, we figured out a method, which can decrease the influence from data amount, which is called "5-folds cross validation". It means that we switch both training and test set after one epoch five times, as shown in Fig 17. We would divide the data set into 5 parts, each time use a different training set and testing set to train the model. Finally, the average of the five results is the desired result. The method is also very common to solve the problem that we wet during training the model.

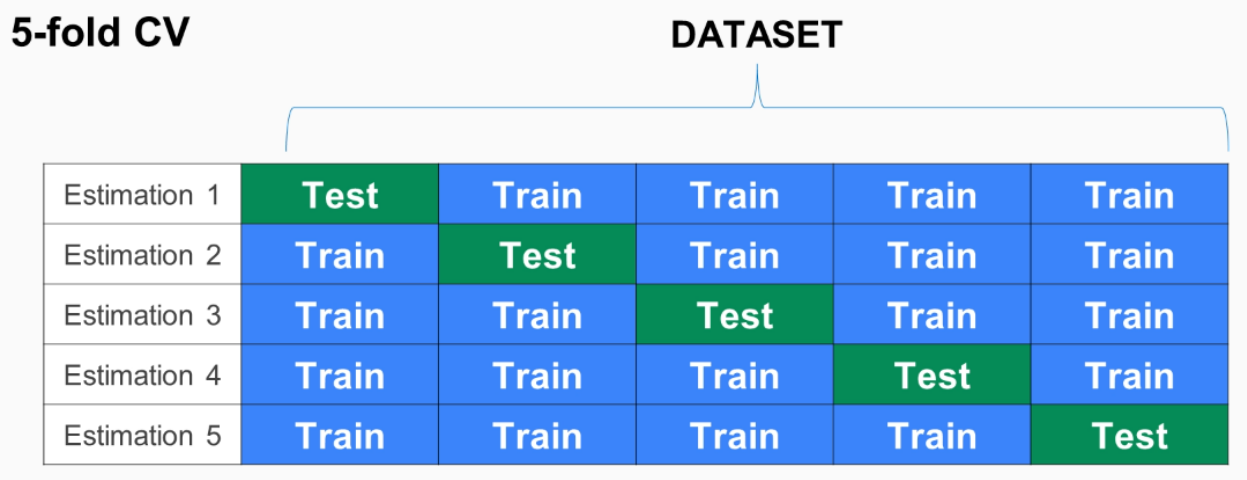


Figure 17. 5-folds cross validation

In the first part of this research, a two-domain confusion matrix can be used to express its network performance. During our research, each fold would have 150 images with the license plate and 150 images without the license plate. We concluded the statistics of the 5-fold cross test, as shown in Table 3 and Table 4.

| Prediction / Times | TP | FP | FN | TN |
|---|---|---|---|---|
| 1-fold | 150 | 0 | 3 | 147 |
| 2-fold | 145 | 5 | 2 | 148 |
| 3-fold | 143 | 7 | 4 | 146 |
| 4-fold | 142 | 8 | 5 | 145 |
| 5-fold | 150 | 0 | 7 | 143 |

Table 3. The results of 5-folds cross validation

| Statistics / Method | fold | Accuracy | Precision | Detection rate |
|---|---|---|---|---|
| YOLOv5 | fold 1 | 99% | 100% | 98.04% |
| | fold 2 | 97.66% | 96.66% | 98.63% |
| | fold 3 | 96.33% | 95.33% | 97.28% |
| | fold 4 | 95.66% | 94.66% | 96.60% |
| | fold 5 | 97.66% | 100% | 95.54% |
| | **Average** | **97.26%** | **97.33%** | **97.22%** |

Table 4. The results of 5-fold cross validation

In Table 3 and 4, we can conclude that we got great performance in 5 folds, and the highest one which can reach 99%. Both precision and detection rate have good performance too. As the data was shown, we can easily conclude that our training model would have great capability to use on license plate recognition systems in the future, because that license plate location is one of the important key points for the whole system. If our training model includes a high score during the license plate location part, we probably would have high chances to get a high score in the digit and letter recognition part. After we finish the score calculation, we also try to use our database on other license plate recognition systems, as shown in Table 5.

| | Accuracy | Precision | Detection rate | Training time for 20 epochs (s) | Database |
|---|---|---|---|---|---|
| YOLOv5 | 97.26% | **97.33%** | **97.22%** | 15:40 | License plate database |
| Faster-RCNN [8] | **98.76%** | 95.48% | 90.30% | 35:16 | |
| Conventional Technology [4] | 84.75% | 72.54% | 70.64% | 0 | |

Table 5. Comparison with other methods

In this comparison, Faster-RCNN has the highest accuracy. But both precision and detection rate is not higher than YOLOv5. It is because we only have one category during the training, and Faster-RCNN has a very heavy network. Those conditions would make the Faster-RCNN too sensitive to detect the license plate so that it would appear to recall frequently. Although faster-RCNN has the highest accuracy, it will also take the longest time to train the model. In conventional technology, both precision and detection have the lowest score, because this method will be affected by many conditions. For example, sunlight and rain are the two major conditions. Although the accuracy of Faster-RCNN is better than our method, YOLOv5 can maintain a good detection rate and Precision. Importantly, our method doesn't need too much time for training.

As we said before, license plate location is one of the important key points for the whole system. If the system can locate the license plate successfully, it is barely to identify the digits and letters on the license plate. Because of the result of license plate location, YOLOv5 also gets the highest score on the digit and letter recognition, as shown in Table 6. If the sensitive problem can be solved, Faster-RCNN probably will get the best performance.

| | YOLOv5 | Faster-RCNN [8] | Conventional Technology [4] |
|---|---|---|---|
| Accuracy | 92.56% | 85.6% | 80.40% |

Table 6. Digit and letter recognition

# V. Conclusion

In this thesis, we publish a new license plate recognition system which includes machine learning. Before I selected this topic to do the research, I always stocked in front of the parking lot entrance. The reason, which causes this problem, is that there is no automatic license plate recognition system in whole public parking lots based on my experience. The previous research is all too old to use, and it couldn't adapt to the realistic situation. That is the reason why I selected the YOLOv5, which has high accuracy and light network, for our system. As a result, YOLOv5 is truly suitable for the license plate recognition system. Based on my knowledge, digit and letter recognition are also implemented by using neural networks before. After we try to use "Tesseract-OCR" in our system, we don't need to train another model only for text recognition. In conclusion, if we can maintain high accuracy in the license plate location part, it isn't necessary to train another model, and also can get the system with great performers.

# References


[1] W. Wang, J. Yang, M. Chen and P. Wang, "A Light CNN for End-to-End Car License Plates Detection and Recognition," in *IEEE Access*, vol. 7, pp. 173875-173883, 2019, doi: 10.1109/ACCESS.2019.2956357

[2] C. -H. Lin and Y. Li, "A License Plate Recognition System for Severe Tilt Angles Using Mask R-CNN," *2019 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2019, pp. 229-234, doi: 10.1109/ICAMechS.2019.8861691

[3] Vincent, Luc (August 2006). "Announcing Tesseract OCR". Archived from the original on October 26, 2006. Retrieved 2008-06-26

[4] R. R. Palekar, S. U. Parab, D. P. Parikh and V. N. Kamble, "Real time license plate detection using openCV and tesseract," *2017 International Conference on Communication and Signal Processing (ICCSP)*, 2017, pp. 2111-2115, doi: 10.1109/ICCSP.2017.8286778

[5] Kay, Anthony (July 2007). "Tesseract: an Open-Source Optical Character Recognition Engine". Linux Journal. Retrieved 28 September 2011.

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, 2016, pp. 770-778.

[7] Y. Shin and I. Balasingham, "Comparison of hand-craft feature based SVM and CNN based deep learning framework for automatic polyp classification," *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, pp. 3277-3280, doi: 10.1109/EMBC.2017.8037556.

[8] X. Mo, K. Tao, Q. Wang and G. Wang, "An Efficient Approach for Polyps Detection in Endoscopic Videos Based on Faster R-CNN, " 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 2018, pp. 3929-3934.

[9] C. -H. Lin and Y. Li, "A License Plate Recognition System for Severe Tilt Angles Using Mask R-CNN," *2019 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2019, pp. 229-234, doi: 10.1109/ICAMechS.2019.8861691

[10] H. Li, P. Wang and C. Shen, "Toward End-to-End Car License Plate Detection and Recognition With Deep Neural Networks," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1126-1136, March 2019, doi: 10.1109/TITS.2018.2847291.

[11] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[12] A. (2021, April 28). *Introduction to optimizers*. Algorithmia Blog.

[13] Shen, K. (2018a, June 20). *Effect of batch size on training dynamics - Mini Distill*. Medium.

[14] Gaillard, F. (n.d.). *Epoch (machine learning) | Radiology Reference Article Radiopaedia.org*. Radiopaedia.

[15] U. (2020). *ultralytics/yolov5*. GitHub.