CAR PLATE DETECTION USING REGION-BASED CONVOLUTIONAL NEURAL NETWORKS

By

SHENGYI LUAN

Presented to the Faculty of Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON
Dec 2020

ABSTRACT

CAR PLATE DETECTION

USING REGION-BASED CONVOLUTIONAL

NEURAL NETWORKS

SHENGYI LUAN

The University of Texas at Arlington, 2020

Supervising Professor: Michael T Manry

Convolutional Neural Networks (CNN) comprise a deep learning technology which is widely used to perform image classification. In this research, we review CNN structures and explain how they can be used for finding license plates in vehicle images. We summarize how the standard CNN processes images into features and compare it to Region-Based Convolutional Neural Networks (R-CNN). After comparing their pros and cons, we decide to design a R-CNN to train our dataset for this project. We find that the one trained with the most training data has the highest testing accuracy. The first training network detector leads to the highest testing accuracy that can reach 0.963 after 10 training epochs. The second training network detector leads to the highest testing accuracy that can reach 0.988 after 20 training epochs.

ACKNOWLEDGEMENTS

I thank my supervising professor Michael T. Manry for his contributions to this project. Dr. Manry guided me and gave me countless suggestions how to organize and proceed with the project. We had many meetings to discuss and solved the problems that we faced.

I would like to thank Dr. Chenyun Pan, Dr. Devarajan Venkat, Dr. Sungyong Jung for taking the time to serve on my thesis committee.

I thank all the members of the lab and friends including Chinmay Rane and Yash Pramod who gave me valuable suggestions on my research.

I would like to thank my family for supporting me during my master's studies in UT Arlington.

TABLE OF CONTENTS

ABSTRACT i

ACKNOWLEGEMENTS iii

LIST OF FIGURES v

CHAPTER I: INTRODUCTION 1

CHAPTER II: DATA 4

A. Available Data 4

B. Vehicle Image Ground Truth Data 6

CHAPTER III: REVIEW OF DEEP LEARNING SYSTEMS 9

A. Multilayer Perceptron (MLP) 9

B. Convolutional Neural Networks 10

C. Region-Based Convolutional Neural Networks 14

CHAPTER IV: METHODOLOGY 18

A. Proposed Algorithm Structure 18

B. Training and Testing Procedures 19

CHAPTER V: RESULTS 23

A. R-CNN Training Results 23

B. R-CNN Testing Results 29

CHAPTER VI: DISCUSSION 31

CHAPTER VII: CONCLUSION 33

REFERENCE 35

LIST OF FIGURES

Figure Page

1. Example Images of Batch1 and Batch10 4

2. Special Cases from Batch1 5

3. Example Images from Batch4 through Batch7 5

4. Example Images from Batch8 and Batch9 6

5. Image Dimensions 7

6. Block Diagram for Generating Ground Truth Data 8

7. Insight of Ground Truth Data in a Table 8

8. MLP Classifier Structure 9

9. Deep CNN Structure 10

10. 2D Filtering Process 11

11. Max Pooling Process 11

12. Possible CNN for Plate Finder 12

13. Selective Search Algorithm 14

14. Generate CNN Features 15

15. Support Vector Machines 15

16. Transformation Between Predicted Bounding Box and Ground Truth Bounding Box 17

17. Block Diagram for Region-Based CNN Plate Detection Design 18

18. All Prediction Values Possible to Locate the Plate 21

19. Intersection over Union 21

20. Training Accuracy and Loss for Detector 1 in 10 Epochs....................................23

21. Training Accuracy and Loss for Detector 2 in 10 Epochs....................................24

22. Training Accuracy and Loss for Detector 3 in 10 Epochs....................................24

23. Training Accuracy and Loss for Detector 4 in 10 Epochs....................................25

24. Training Accuracy and Loss for Detector 5 in 10 Epochs....................................25

25. Training Accuracy and Loss for Detector 6 in 10 Epochs....................................26

26. Training Accuracy and Loss for Detector 7 in 10 Epochs....................................26

27. Training Accuracy and Loss for Detector 8 in 10 Epochs....................................27

28. Training Accuracy and Loss for Detector 9 in 10 Epochs....................................27

29. Training Accuracy and Loss for Detector 10 in 10 Epochs..................................28

30. Two Different Training Network Parameters Setting on 5000 Testing Image......29

31. YoLo Training Dataset 32

CHAPTER I

INTRODUCTION

Artificial intelligence (AI) [33] refers to the simulation of human intelligence processes with computer systems. Many AI applications such as expert systems [47], natural language processing (NLP) [48], speech recognition [50] and machine vision [49] are widely used in science, business and medicine. Machine vision or computer vision is a field of study enabling computers to analyze images or videos [2], so that computers can understand the content in the images or videos. Unlike in human vision, modern deep learning systems need to be trained with many layers before they can perform image recognition and classifications [9]. One popular application of computer vision is in self-driving cars [38]. A computer vision system uses camera video to detect roads, traffic lights and pedestrians. In image classification for disease diagnosis, for example, the computer is trained with thousands of optical coherence tomography (OCT) scans from many eye disease patients [42]. Such a system can detect eye disease efficiently and provide the doctor with enough patient disease information to analyze. In [13], a premise of image classification is that a huge dataset required for training, a system before it is able classify data into different classes.

In the past, the multilayer perceptron (MLP) [37] has been used in computer vision. MLPs typically include three layers that are an input layer, a hidden layer and an output layer. MLPs with a single hidden layer have the universal approximation property [22] and can approximate continuous functions and their derivatives, degree and density function [26] [27]. However, the MLP is not sufficient for modern computer vision because it has a large number of parameters and lacks shift invariance.

In modern computer vision, automatic license plate recognition (ALPR) systems are used for detecting stolen vehicles, toll control and parking lot access [36]. Behind these scenes, CNNs play an important role because they comprise many neurons and have trainable weights and biases [4]. The CNN is mainly used in image processing and classification [39] where it has good performance [51] in object detection [52], and semantic segmentation [19]. Because it can learn high-level features from labeled training data, the CNN can achieve high recall and precision rates in license plate (LP) detection [18]. For object detection, image segmentation [29] is an important first step which generates sub-images of one object at a time. It helps divide images into many small parts that can be individually classified in applications such as face recognition and plate number identification like blob extraction-based character segmentation [66] [16]. The segmentation finds objects and draws rectangles around replacement classifier.

Image segmentation is combined with a CNN in the Region-based Convolutional Neural Network (R-CNN). It comprises three steps which are selective search, CNN feature extraction and SVM classifier for classification after the feature selection [43]. The selective search segments objects in an image after many iterations based on color, texture, and size [25]. The CNN extracts specific features and the SVM classifies each object [37].

In [31], Rafique uses an SVM and R-CNN to detect license plates. The R-CNN selects regions to reduce the search space and detection time as compared to a stand-alone CNN method. Rafique found that Fast-RCNNs and Faster-RCNNs could be used in real time performance. Li *et al.* [20] used a unified method that contains a region proposal network (RPN) and Regions of Interest (ROIs). After a vehicle image goes through the CNN layers, convolutional features are produced. The RPN [17] has a classifier and a regressor. The classifier makes sure the probability of the proposed region has the detected object and the regressor adjusts bounding box coordinates. In the end, they relocate the coordinates of the bounding box and make sure it includes the detected object in the box.

In this project, we use the basic R-CNN to build up a plate detection system which is similar with Rafique's. In chapter II, we describe training and testing datasets, and introduce the size and format of the vehicle images. In chapter III, we review the MLP, CNN, and R-CNN and analyze their structures. In chapter IV, we describe the R-CNN method and use it to design a plate detection system. In chapter V, we demonstrate the training and testing results. In chapter VI, we summarize our work and the remaining problems.

CHAPTER II

DATA

In this chapter, we describe the raw data used for training an R-CNN plate finder and describe how to put it into a useful intermediate form. Then we describe a method for generating ground truth data for the training process.

A. Available Data

There are ten batch folders and each of them has one thousand vehicle images. All of them are in RGB form with 2048x512 pixels in the JPEG format. The images are ranked by image quality from Batch1 to Batch10. In Batch1, the majority of the vehicle images are very clear, and the plates have very good contrast with the car body in each picture. However, in Batch10, most of the images are blurred and hard to make out. Some of them have poor contrast between the background and vehicle body. The images in Fig1 illustrate the difference between Batch1 and Batch10 folder images.

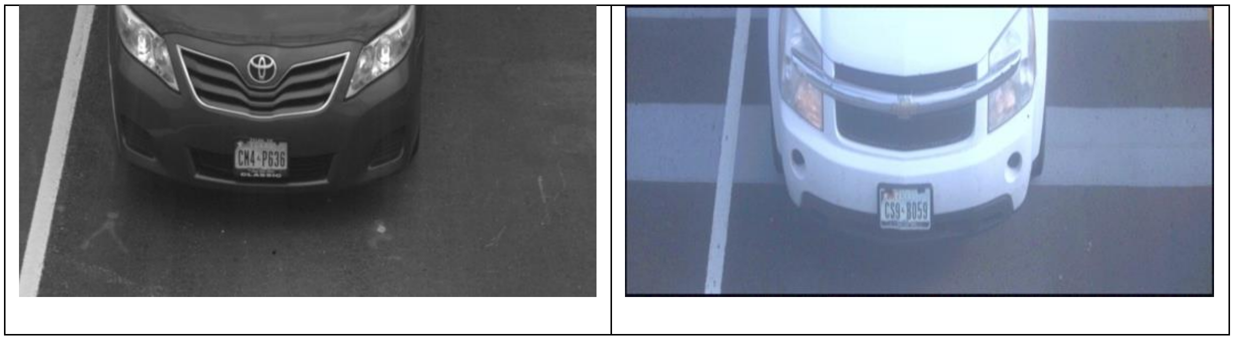


Fig 1. Example Images of Batch1(Left) and Batch10(Right)

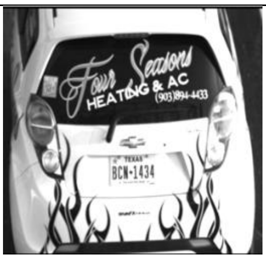
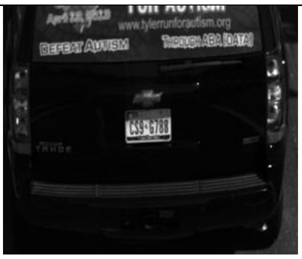
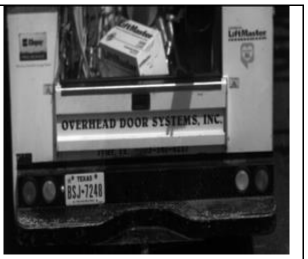
			

Fig 2. Special Cases from Batch1

In the Batch1 images of Fig 2, some images have company stickers and logos near the plate, and a few cars have fancy decorations on their rear windows. Meanwhile, some of the truck images have the plate either at the left bottom corner or right bottom corner of the vehicle. The Batch2 and Batch3 folders both have similar image quality. A few images have bright headlights making it difficult to read the plates.

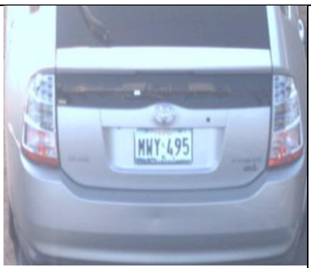
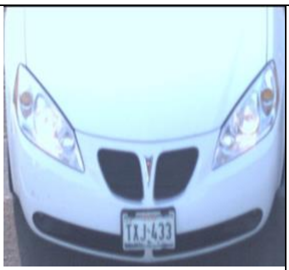
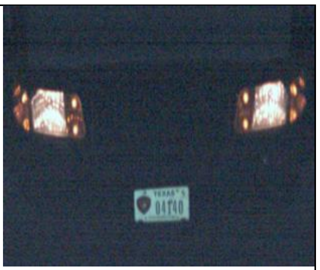
			

Fig 3. Example Images from Batch4 through Batch7

Fig 3 shows vehicle images from Batch4 through Batch7 which have lower quality. Many of them were snapped at night and have unclear plates.

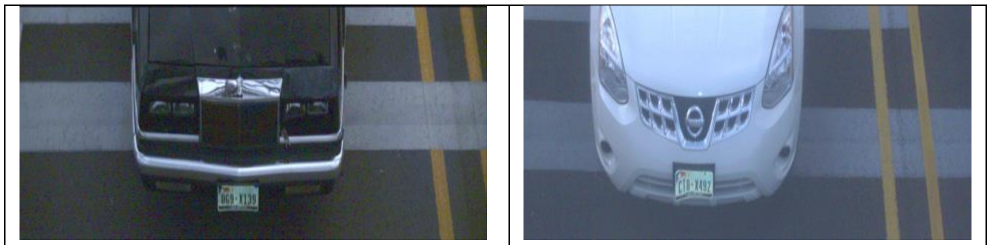


Fig 4. Example Images from Batch8 and Batch9

Fig 4 shows that Batch8 and Batch9 images are similar to the low quality Batch10 images with many road and pedestrian lanes visible near the vehicles. Many vehicle images taken at night are more blurred with hard to read plates.

B. Vehicle Image Ground Truth Data

The ground truth consists of the coordinates of the plates' upper left-hand and lower right-hand pixels. The coordinates can be compared those of a plate finder to test its accuracy, as we shall see later. There are two classes to be distinguished, plate and non-plate, so we usually need to label plate pixels and assume the others are in the non-plate class. Each vehicle image corresponding to x- axis and y-axis as shown by in Fig 5 when the MATLAB load each of them for collecting ground-truth data.

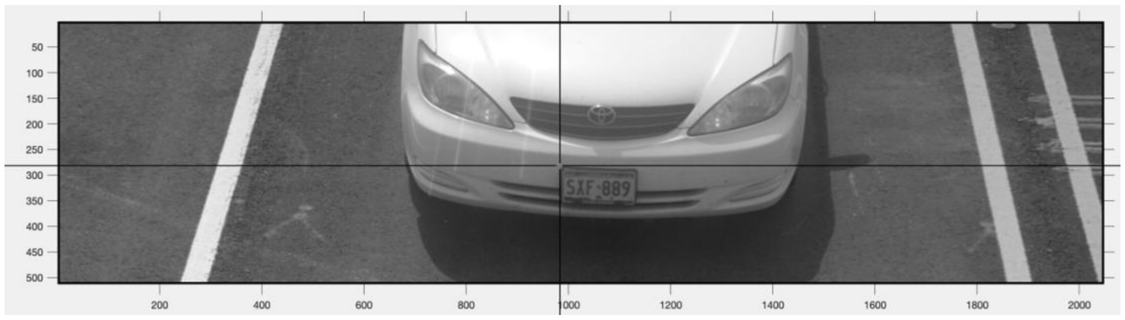


Fig 5. Image Dimensions [8]

The bounding box coordinates are x_1, y_1, w, h [8]. The width and height are calculated as

$$w=x2-x1$$
$$h=y2-y1 \tag{2.1}$$

where (x1, y1) are the coordinates of a plate's left top corner and (x2, y2) are the coordinates of a plate's right bottom corner.

For the thousands of vehicle images, we use a loop to load them one by one, so the user can indicate the ground truth. The block diagram in Fig 6 illustrates how the ground truth data is generated for a batch folder

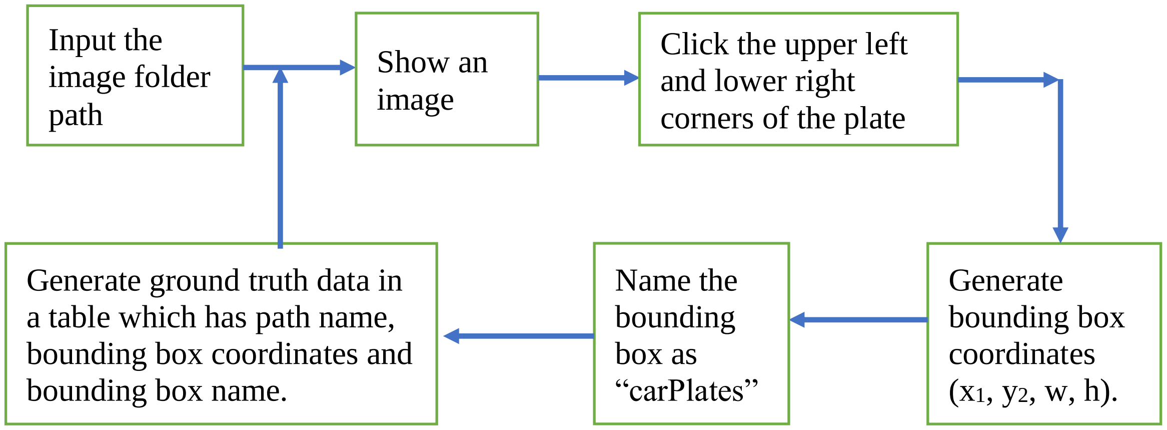
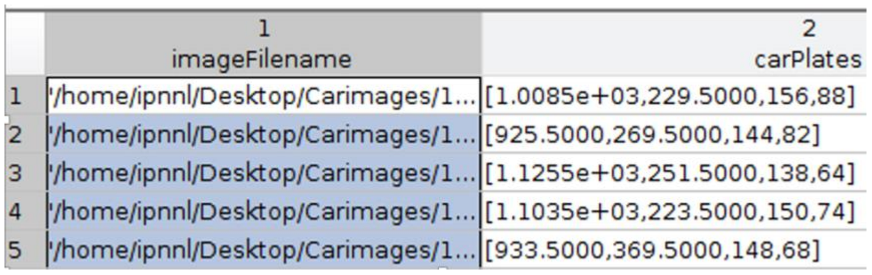

Input the image folder path
Show an image
Click the upper left and lower right corners of the plate
Generate bounding box coordinates (x_1, y_2, w, h).
Name the bounding box as "carPlates"
Generate ground truth data in a table which has path name, bounding box coordinates and bounding box name.

Fig 6. Block Diagram for Generating Ground Truth Data

After vehicle images are processed, the ground truth dataset consists of image path names, bounding box data and bounding box name as "carPlates". The Fig 7 shows the ground truth data format that generate in a table.



	1 imageFilename	2 carPlates
1	'/home/ipnnl/Desktop/Carimages/1...	[1.0085e+03,229.5000,156,88]
2	'/home/ipnnl/Desktop/Carimages/1...	[925.5000,269.5000,144,82]
3	'/home/ipnnl/Desktop/Carimages/1...	[1.1255e+03,251.5000,138,64]
4	'/home/ipnnl/Desktop/Carimages/1...	[1.1035e+03,223.5000,150,74]
5	'/home/ipnnl/Desktop/Carimages/1...	[933.5000,369.5000,148,68]

Fig 7. Insight of Ground Truth Data in a Table

CHAPTER III

REVIEW OF DEEP LEARNING SYSTEMS

In this chapter, we explore the multilayer perceptron (MLP)'s structure in section A, the CNN structure in section B, and the R-CNN structures in section C. We discuss how each of them processes images to perform feature extraction, and the reason we choose R-CNN to design this project.

A. Multilayer Perceptron (MLP)

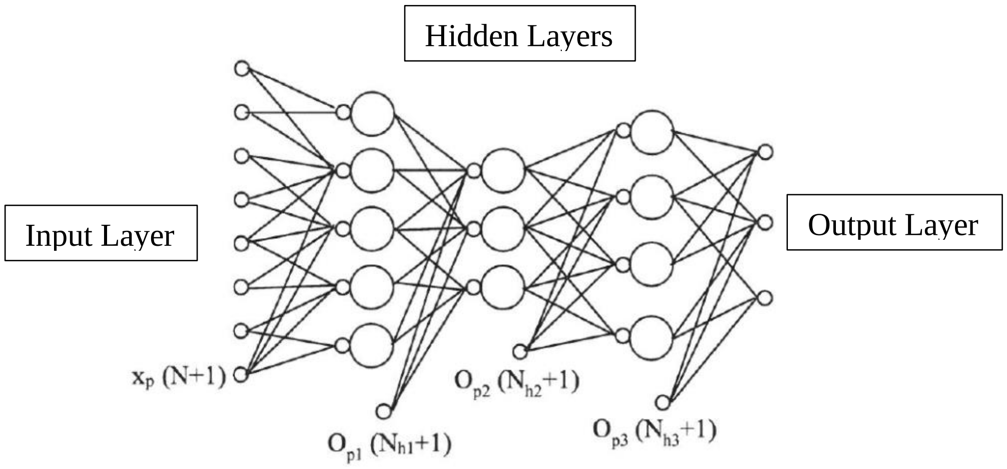


Fig 8. MLP Classifier Structure [10]

The MLP uses nonlinear basis functions in the hidden units [1] and can have many hidden layers between an input layer and an output layer that are fully connected. In [1], each input variable dots product with the weights and biases to form activations. Then the activations process the MLP's hidden layer nonlinear activation functions like Gaussian, ReLU, Sigmoid with into output activations. The output activations go through activation function to generate the output layer. A backpropagation algorithm can be applied to the activation function to do classification [22].

In [26][27], The Weierstrass theorem [61] is used to prove that a single hidden layer MLP has the universal approximation property. This means that any continuous function over a bounded compact input space can be approximated by the MLP [27]. One benefit of universal approximation is that Bayes discriminant functions for any number of classes can be approximated arbitrarily well [5].

B. Convolutional Neural Networks

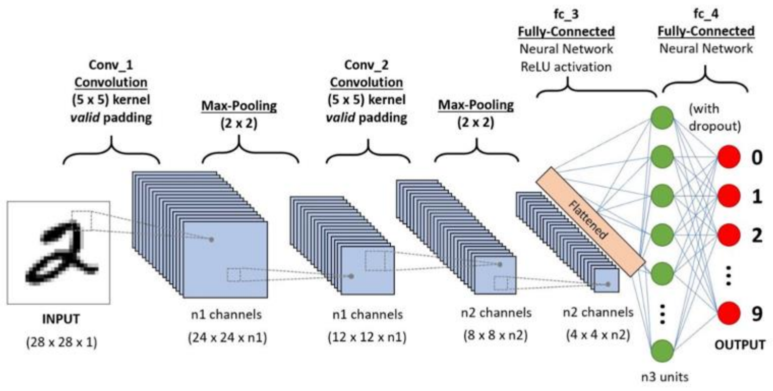


Fig 9. Deep CNN Structure [34]

In Fig 9, the CNN [55] [56] [57] [58] has several types of layers including convolution layers, nonlinear activation layers, pooling layers, final feature layers and a classification layer. The convolutional layer consists of coefficients connecting an input image layer to an output image layer. It can process rectangular blocks of input image pixels into feature pixels in the following layer.

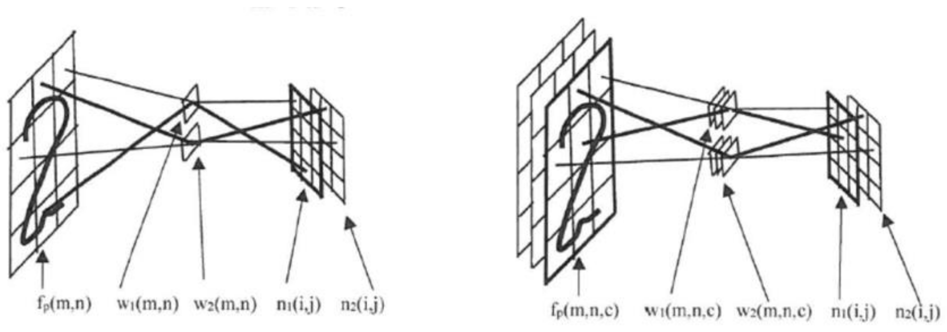


Fig 10. 2D Filtering Process [10]

An activation layer applies nonlinear activations to a net function layer which is linearly dependent upon the previous activation layer. There are three different activation functions used in CNNs including linear activations, softmax activations and Rectified Linear Unit (ReLU) activations [10].

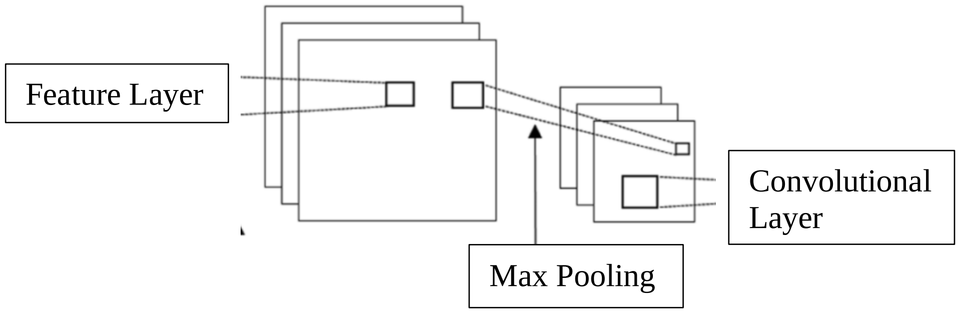


Fig 11. Max Pooling Process

The pooling layers use either max pooling or average pooling to reduce the size of a nonlinear feature layer. Max pooling outputs the maximum pixel value over a rectangular window. In contrast, average pooling outputs the average pixel value over the window.

The final feature layer is located just before the classifier and the number of features N_F can be calculated as

$$N_F = K \cdot M_o \cdot N_o \tag{3.1}$$

where K is the number of filters and M_o and N_o are the input image's dimensions

In the classification layer, there are several possible output layer activations including sigmoid, softmax and ReLU functions. Each of them has different responses to net function outliers and biases. Here we used ReLU function in object detection.

There are some drawbacks with using CNNs. For example, when some features are fed to the classifier, each feature is a function of a specific block from the previous layer or image. Each block is constrained to have the same feature definitions as its neighbors. In general, the CNN lacks shift invariance when the detectable object is small [53].

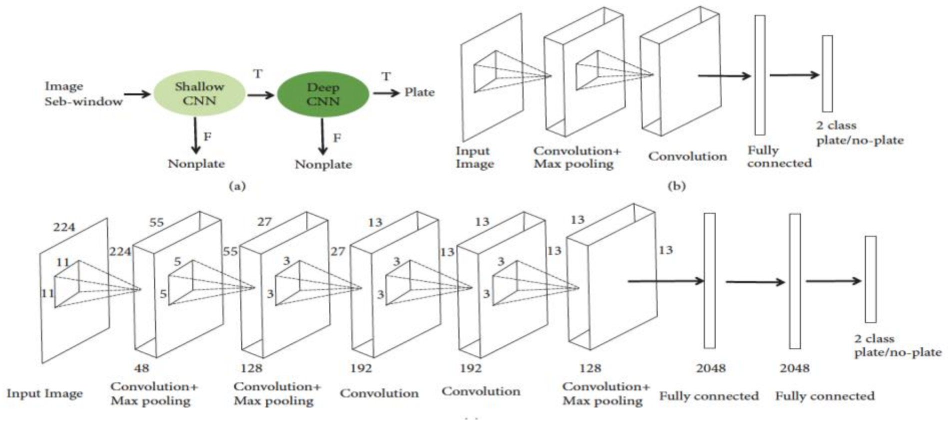


Fig 12. Possible CNN for Plate Finder [54]

A plate finder could be designed by using one shallow CNN and one deep CNN that both classify plate or non-plate feature blocks [54]. The shallow CNN includes four layers and is used to quickly scan the image and remove most background regions. Then the deep CNN can be used to perform plate detection using the remaining regions. In [54], Zou *et al* proposed that a window be used to scan the vehicle image and generate possible candidate windows in a shallow CNN. In their experiment, after the shallow CNN, there were 20 possible candidates left for the deep CNN to classify and they could reach 98.1% accuracy on testing dataset.

In [29], Sharma claims that using the CNN to classify objects in an image requires too much computation. First of all, the image will be divided into many small blocks, and the CNN classifies them into different classes. Then all the resulting class blocks in a given class are combined into a detectable object region. Due to the different objects own different shape and size, the CNN will take too much computation time. To solve computational and object localization problems, Region-Based Convolutional Neural Networks will be introduced.

C. Region-Based Convolutional Neural Networks

In this subsection, we review the R-CNN structure and describe how it generates and processes bounding boxes in an image.

1. Selective Search Algorithm

In the R-CNN, using a Selective Search Algorithm (SSA) in object detection [12] [54] to generate the bottom-up region proposals [14]. The SSA [62] begins with an input image, the input image becomes over segmented into many small regions with different color and size. Then the SSA merges these small regions into larger regions that base on any two small regions have similar color, texture, size and shape. The SSA also adds bounding box to each proposal region after first calculation and repeats this process with the larger regions again and again until the top image. In the whole process, the SSA forms region proposals and possible region proposals on the top of the image in Fig 13. In [62], Chandel said that usually 1000 to 1200 proposals could be good enough to locate all the objects correctly in an image.

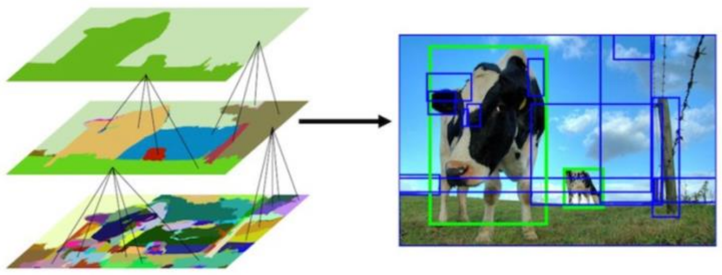


Fig 13. Selective Search Process [62]

2. CNN Features Extraction

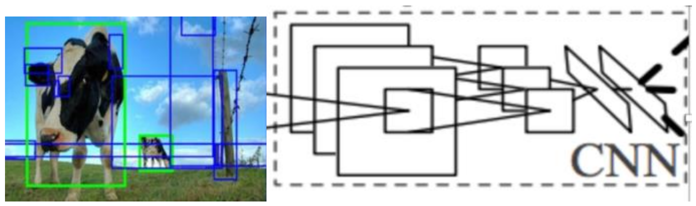


Fig 14. Generate CNN Features [14]

After the SSA produces region proposals, each of this region proposal goes through a convolutional network. The CNN extracts a fixed length feature vector when each region proposals go through the network.

3. Support Vector Machines (SVM)

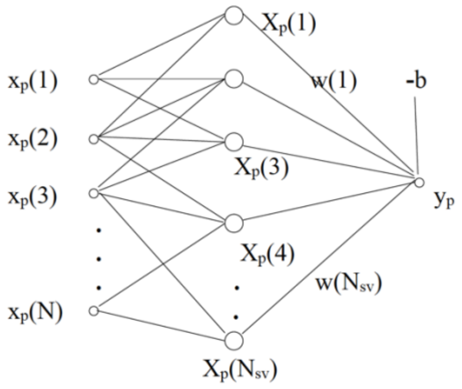


Fig 15. Support Vector Machines [10]

When the CNN feature vectors go to a linear support vector machines (SVM), the SVM classifies these CNN feature vectors [14] [59] into different classes. The SVM gives the correct object with its bounding box that depends on the dataset used for training.

4. Bounding Box Regressor

A bounding box regressor is used to adjust a predicted bounding box [64] [65] to reduce its localization errors. The predicted bounding box coordinates are [43]:

$$\boldsymbol{P} = (P_x, P_y, P_w, P_h) \tag{3.2}$$

where P_x, P_y are the center point coordinate of the predicted bounding box and P_w, P_h are the width and height of predicted bounding box

The regressor tries to adjust the predicted bounding box near to a ground truth bounding box. The regressor function is showed and perform these operations.

A ground truth bounding box coordinate is [43] denoted as:

$$\hat{\boldsymbol{g}} = (\hat{g}_x, \hat{g}_y, \hat{g}_w, \hat{g}_h) \tag{3.3}$$

where $\hat{g}_x$ and $\hat{g}_y$ are the center point coordinate of ground truth bounding box, $\hat{g}_w$ and $\hat{g}_h$ are the width and height of detected bounding box

$$\begin{aligned} \hat{g}_x &= p_w d_x(\mathbf{p}) + p_x \\ \hat{g}_y &= p_h d_y(\mathbf{p}) + p_y \\ \hat{g}_w &= p_w \exp(d_w(\mathbf{p})) \\ \hat{g}_h &= p_h \exp(d_h(\mathbf{p})) \end{aligned} \tag{3.4}$$

where $d_x(\mathbf{p})$ is the horizontal distance between detected bounding box central point (P_x, P_y) and ground truth bounding box central point ($\hat{g}_x$, $\hat{g}_y$). $d_y(\mathbf{p})$ is the vertical distance between the detected bounding box central point (P_x, P_y) and the ground truth bounding box central point ($\hat{g}_x$, $\hat{g}_y$). $d_w(\mathbf{p})$ is the shifting distance between the detected bounding width and the ground truth bounding box width. $d_h(\mathbf{p})$ is the shifting distance between the detected bounding bosx and the ground truth bounding box height.

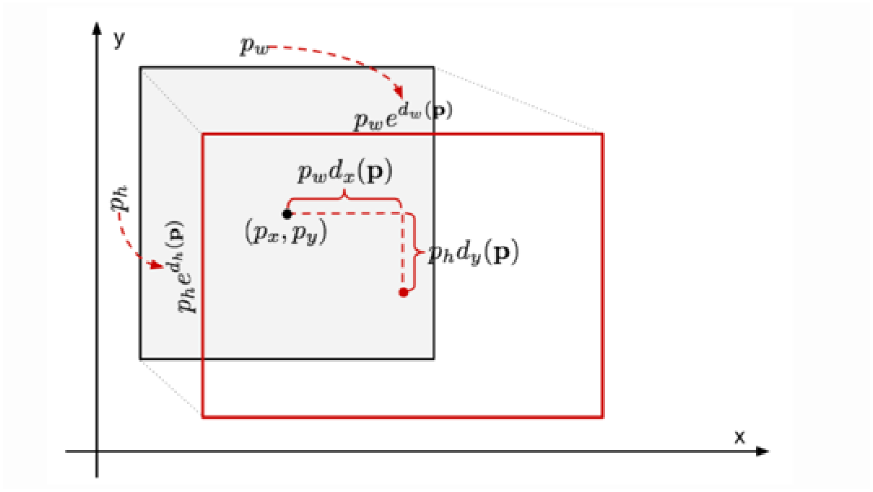


Fig 16. Transformation Between Predicted Bounding Box and Ground Truth Bounding Box [43]

The Region-Based CNN (R-CNN) is used for plate detection because it can reduce computational time when an object's size in an image, are different [29]. R-CNN's selective research will help to generate object outline instead of using CNN only to sperate the object into blocks and combine them in the end when detected. In addition, the R-CNN uses a selective search process to automatically generate 2000 region proposals, so it is able to locate the most important objects. The R-CNN [39] uses variable size sliding windows and sends the windows to a CNN to get features vectors and uses an SVM to classify their corresponding objects. Alternate versions of a R-CNN include the fast-RCNN, faster-RCNN and mask-RCNN, each with different training speed and testing accuracy.

CHAPTER IV

METHODOLOGY

In this chapter, we show a project design flow chart in section A, then we discuss about training and testing procedures for this project in section B.

A. Proposed Algorithm Structure

In this subsection, we show the design block diagram:

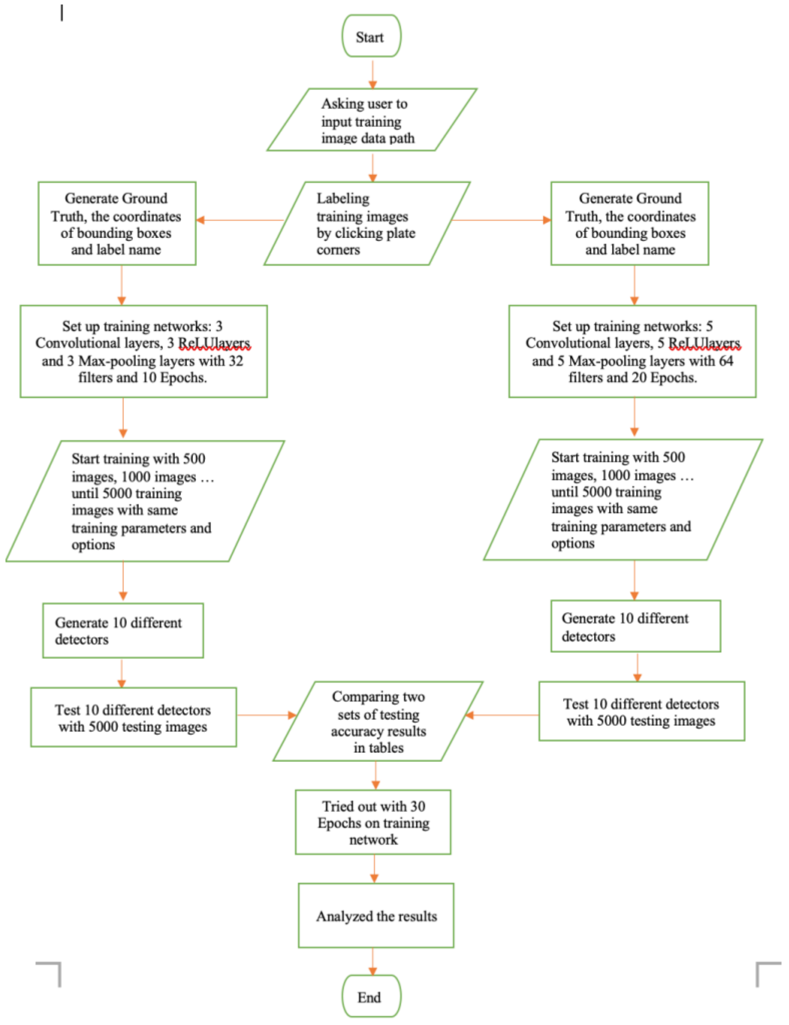

Start
Asking user to input training image data path
Labeling training images by clicking plate corners
Generate Ground Truth, the coordinates of bounding boxes and label name
Generate Ground Truth, the coordinates of bounding boxes and label name
Set up training networks: 3 Convolutional layers, 3 ReLUlayers and 3 Max-pooling layers with 32 filters and 10 Epochs.
Set up training networks: 5 Convolutional layers, 5 ReLUlayers and 5 Max-pooling layers with 64 filters and 20 Epochs.
Start training with 500 images, 1000 images … until 5000 training images with same training parameters and options
Start training with 500 images, 1000 images … until 5000 training images with same training parameters and options
Generate 10 different detectors
Generate 10 different detectors
Test 10 different detectors with 5000 testing images
Comparing two sets of testing accuracy results in tables
Test 10 different detectors with 5000 testing images
Tried out with 30 Epochs on training network
Analyzed the results
End

Fig 17. Block Diagram for Region-Based CNN Plate Detection Design

Fig 17 shows the project progress in our experiment. We trained our training datasets with two different training networks and parameters, then we compare their testing accuracies.

B. Training and Testing Procedures

For each batch folder the images are randomly separated into 500 training images and 500 testing images. We trained 10 different R-CNNs plate detectors. Detector1 has 500 training images, Detector2 has 1000 images, Detector3 has 1500 images etc. Each detector has 500 more training images then the previous one and Detector10 has 5000 training images from Batch1 through Batch10.

In the R-CNN’s CNN, we set up and compared two different sets of training parameters. We used 3 convolutional layers, 3 max-pooling layers and 3 ReLU layers with a stride of 1 for the feature learning process and the filter kernel size is 5x5 with 32 filters in this project with 10 epochs at first. The second set of training parameters that we used 5 convolutional layers, 5 max-pooling and 5 ReLU layers, the filter kernel size is 5x5 with 64 filters with 20 epochs. A feature vector was generated by the convolutional layers, then the SVM showed an accuracy value to confirm the bounding box locate the plate correctly. If there no plate is detected, the SVM showed a ‘0’ accuracy value without any bounding box showed.

The initial learning rate ranged from 0 to 1 and the minimum learning rate was 10^{-6} (LR) in MATLAB. The smaller learning rate will take more training time and the minibatch size was 10. We allowed each detector to have the same parameter settings and summarized their testing performances in tables.

During training by stochastic gradient descent momentum (SGDM) [3], the algorithm groups the full dataset into disjoint mini batches. Each iteration corresponds to the calculation of the network's gradient for each mini batch. Each epoch corresponds to moving through every available mini batch.

Stochastic gradient descent momentum performs the operations

$$V_t = \beta V_{t-1} + (1-\beta)\nabla_w L(W, X, y)$$
$$W = W - \alpha V_t \tag{4.1}$$

where $\mathbf{V_t}$ is the sequence of matrix and $\mathbf{V_{t-1}}$ is the previous sequence matrix, ß is the hyper parameter from 0 to 1, L is Loss Function, ∇_w is the weight matrices gradient matrices, α is the learning rate, **W** is the weight matrix, **X** is the training input mini batch image matrix, and **y** is the training output mini batch image matrix [3].

The training loss mini batch accuracy uses the cross-entropy Loss Function:

$$L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -\sum_i \boldsymbol{y}_i \ln(\widehat{\boldsymbol{y}}_i) \qquad [40]\ (4.2)$$

where i is the number of minibatches, $\widehat{\boldsymbol{y}}$ is the predicted probability vector and $\boldsymbol{y}$ is the ground truth vector.

After the fixed size feature vector was generated by the convolutional neural network, the bounding regressor and SMV to classify the plate. Meanwhile, the training accuracy and loss were defined as mini training batch accuracy and mini training batch loss at each epoch in the training process, they were generated into graphs.

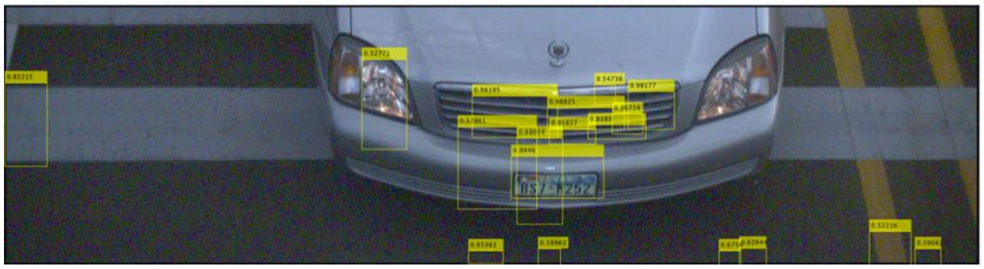


Fig 18. All Prediction Values Possible to Locate the Plate

In determining if the plate is found correctly during testing process, many possible plate regions were detected in each testing vehicle image and one example showed in Fig 18 with more than ten possible bounding boxes were detected. How the SVM determine which possible plate region is correct that based on the Intersection over Union (IoU) value.

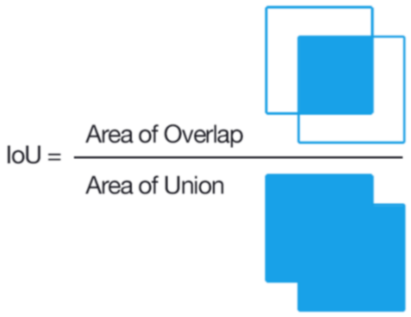


Fig 19. Intersection over Union

This IoU value is calculated by area of overlap region between the possible bounding box region and the ground truth bounding box region dividing by total area of two regions [60]. All the predicted regions with IoU values which higher than 0.5 were shown with bounding boxes [59]. The rest will be ignored. After the SVM calculate each possible plate regions with different IoU values, the SVM will select the highest IoU value region as the detected plate region [23]. When no IoU value is higher than 0.5, the SVM does not detect the plate and no bounding box showed in the image.

In the last, the bounding box regressor will tight up the highest score plate bounding box to be more accurate [23]. Each of 5000 testing images shows a bounding box location, a prediction value and an object label name in the testing process. We are going to use ten detectors and each of them is going to test with all 5000 testing images.

CHAPTER V

RESULTS

In this chapter, we show the R-CNN training results with each of the ten detectors and summarize testing accuracy results from each detector for the 5000 testing images of section B.

A. R-CNN Training Results

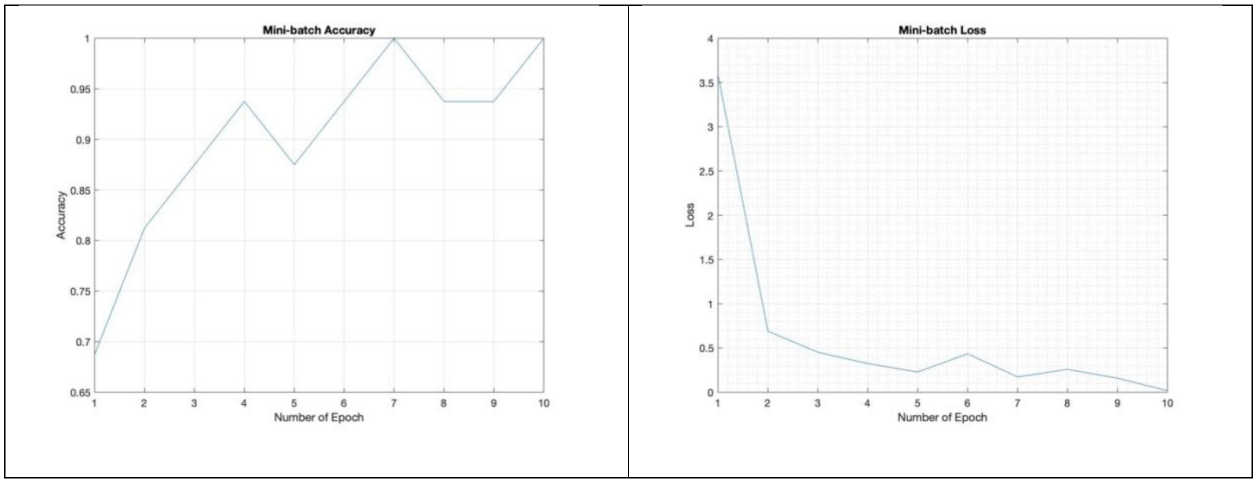


Fig 20. Training Accuracy and Loss for Detector 1 in 10 Epochs

The Detector1 training accuracy starts at 0.68 and reaches an average of 0.85 after the 3rd epoch. The loss dropped from 3.5 to less than 0.5 after the 3rd epoch.

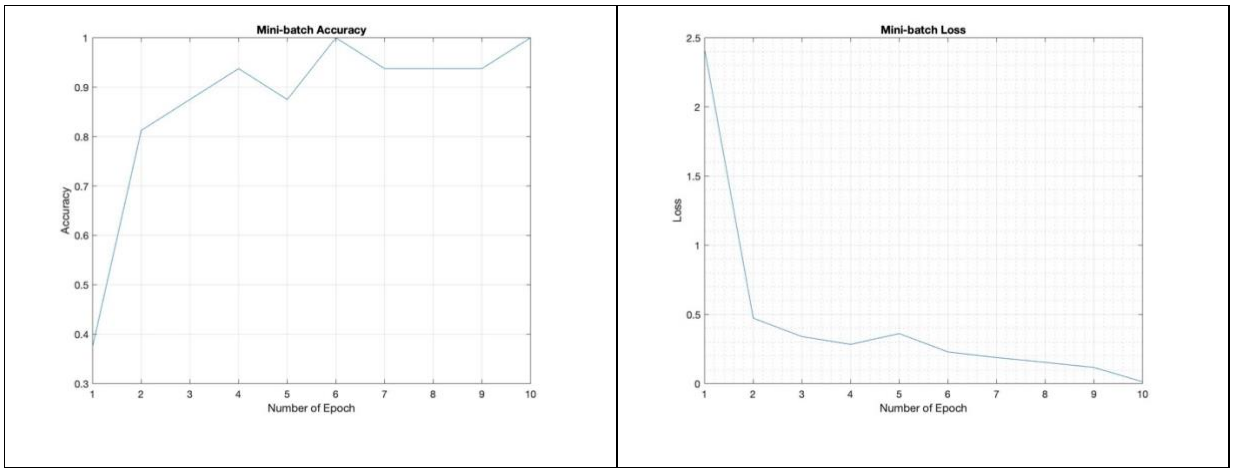


Fig 21. Training Accuracy and Loss for Detector 2 in 10 Epochs

The Detector2 training accuracy began at 0.37 and it increased to 0.8 after the 2nd epoch. The loss dropped from 2.5 to less than 0.5 after the 2nd epoch.

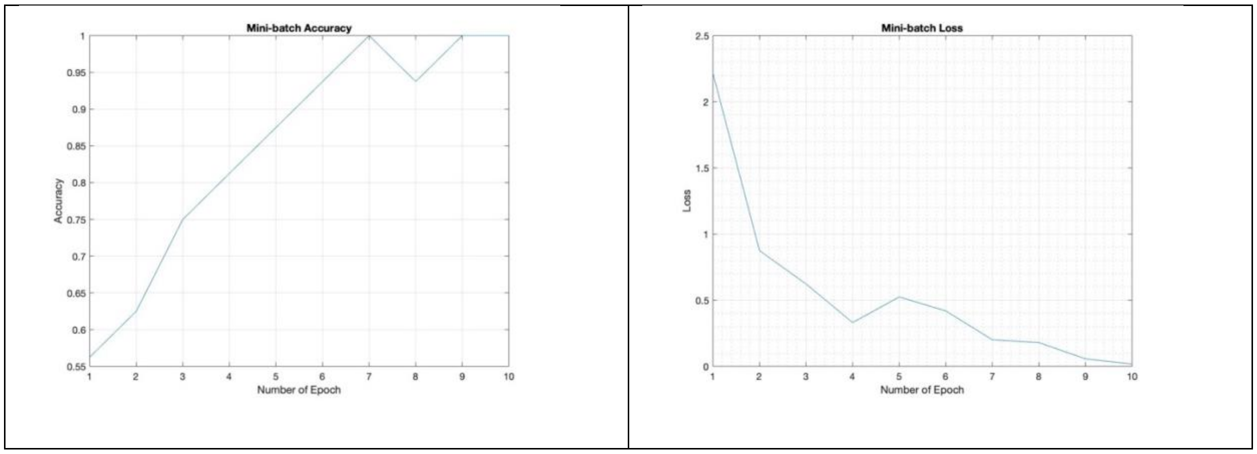


Fig 22. Training Accuracy and Loss for Detector 3 in 10 Epochs

The Detector3 training accuracy began at 0.55 and it increased to about 0.8 after the 4th epoch. The loss dropped from 2.3 to less than 0.5 after the 4th epoch.

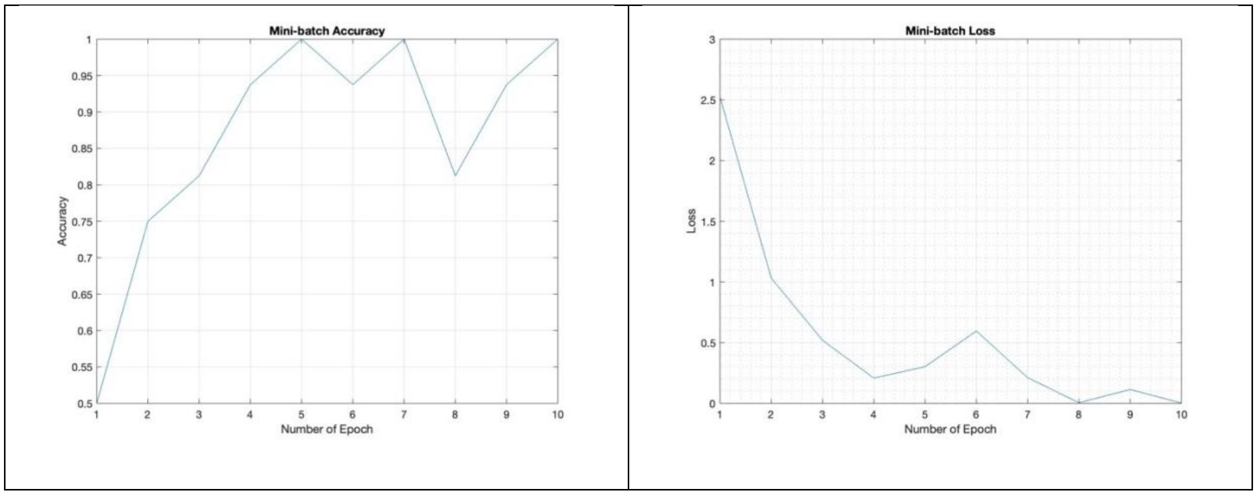


Fig 23. Training Accuracy and Loss for Detector 4 in 10 Epochs

The Detector4 training accuracy began at 0.5 and it increased to about 0.8 after the 3rd epoch. The loss dropped from 2.5 to less than 0.5 after the 3rd epoch.

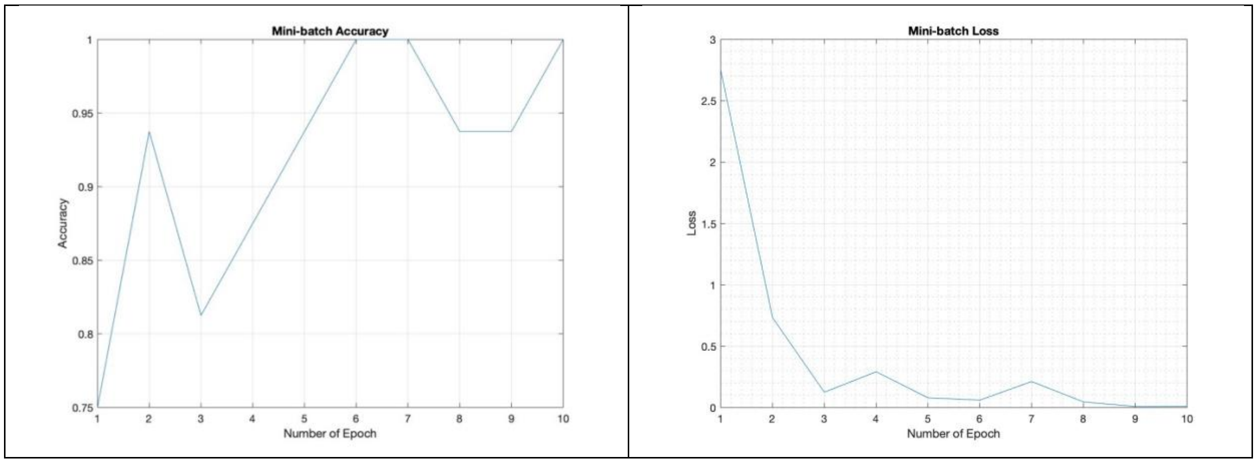


Fig 24. Training Accuracy and Loss for Detector 5 in 10 Epochs

The Detector5 training accuracy began at 0.75 and it increased about 0.8 after the 1st epoch. The loss dropped from 2.7 to less than 0.5 after the 2nd epoch.

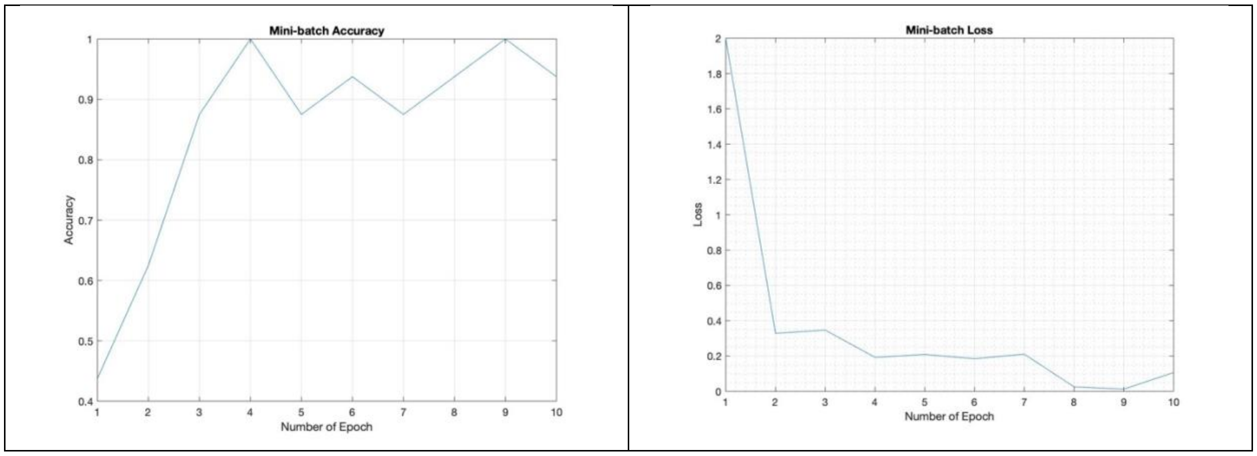


Fig 25. Training Accuracy and Loss for Detector 6 in 10 Epochs

The Detector6 training accuracy began at 0.45 and it increased about 0.8 after the 3rd epoch. The loss dropped from 2 to less than 0.4 after the 2nd epoch.

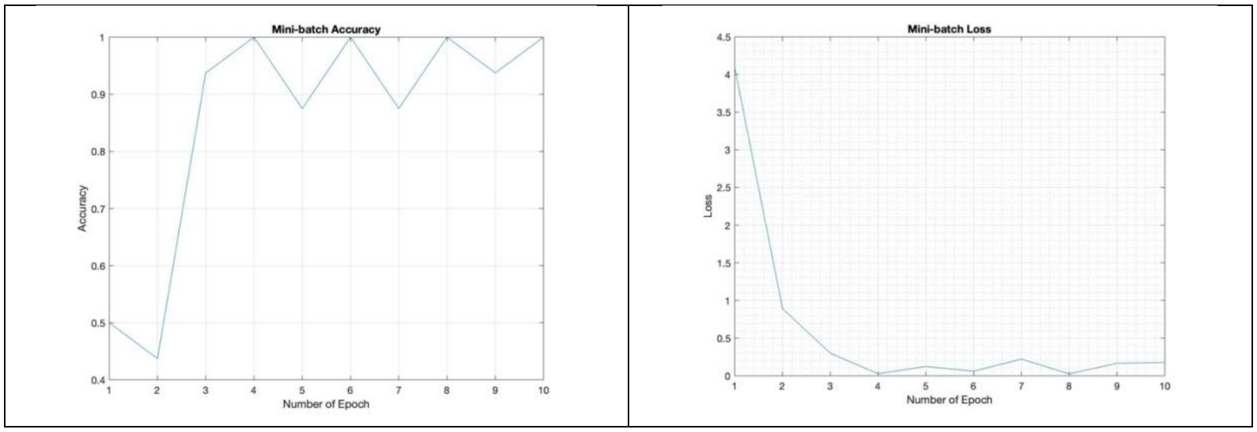


Fig 26. Training Accuracy and Loss for Detector 7 in 10 Epochs

The Detector7 training accuracy began at 0.5 and it increased about 0.8 after the 3rd epoch. The loss dropped from 4 to less than 0.5 after the 3rd epoch.

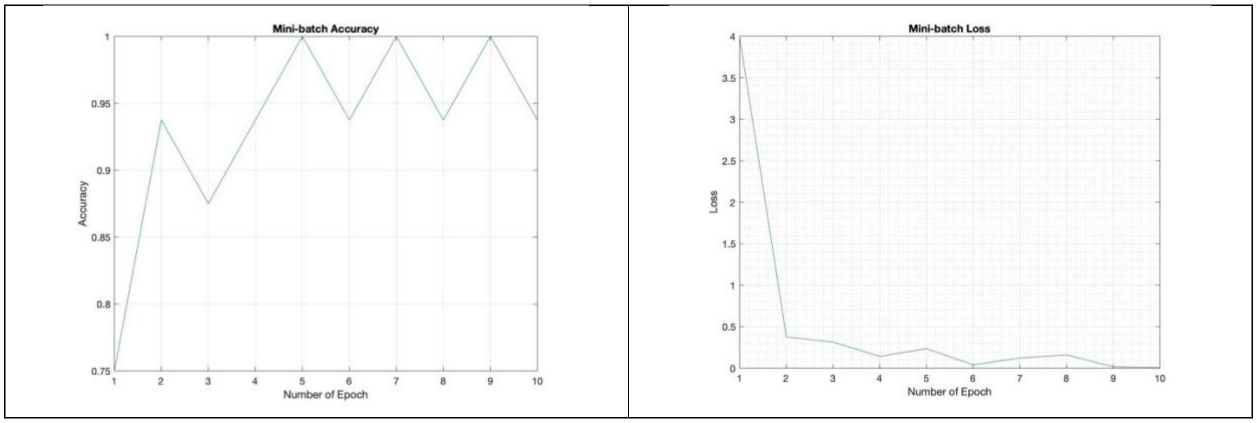


Fig 27. Training Accuracy and Loss for Detector 8 in 10 Epochs

The Detector8 training accuracy began at 0.75 and it increased about 0.8 after the first epoch. The loss dropped from 4 to less than 0.5 after the 2nd epoch.

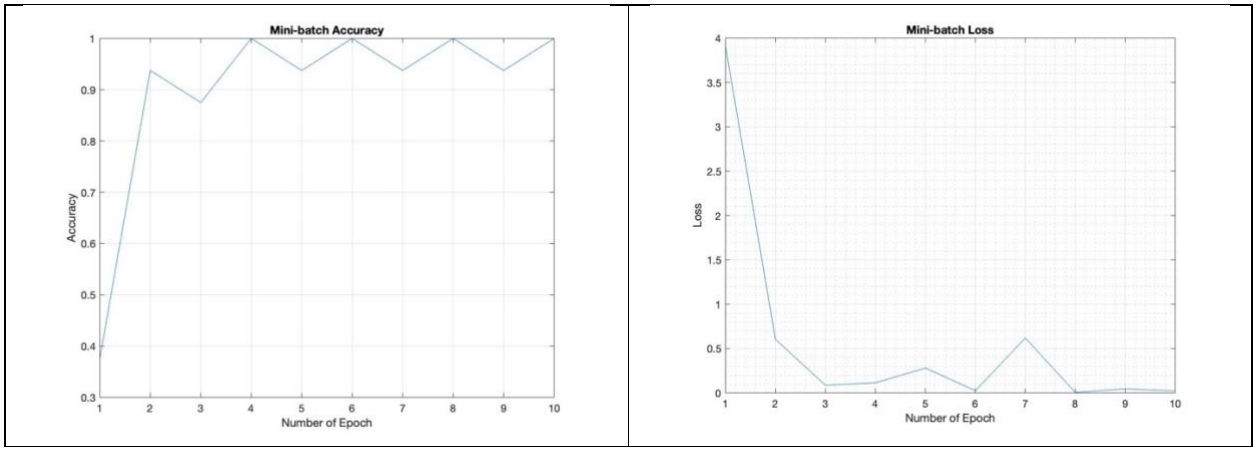


Fig 28. Training Accuracy and Loss for Detector 9 in 10 Epochs

The Detector9 training accuracy began at 0.37 and it increased about 0.8 after the first epoch. The loss dropped from 4 to less than 0.5 after the 2nd epoch.

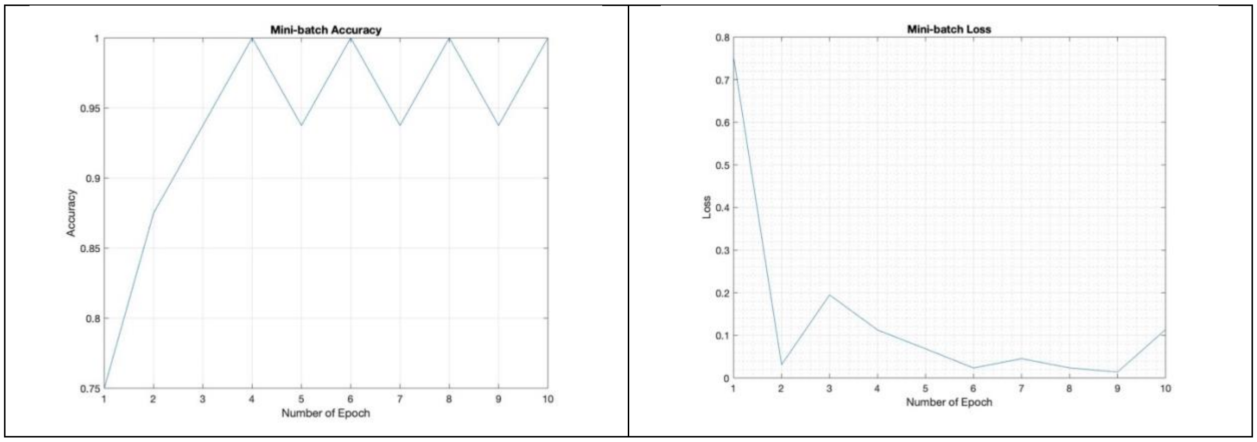


Fig 29. Training Accuracy and Loss for Detector 10 in 10 Epochs

The Detector10 training accuracy began at 0.75 and it increased about 0.8 after the 2nd epoch. The loss dropped from 0.8 to less than 0.1 after the 2nd epoch.

From these ten detector training results, Detector10 showed that the average accuracy started from 0.75 and reached more than 0.9 after the 10th epoch. The loss dropped from 0.78 to average below 0.1 after the 4th epoch. Detector10 showed better training results than the others, and its testing accuracy reached its highest value after the 8th epoch.

B. R-CNN Testing Results

Network	**1st Testing Accuracy**	**2nd Testing Accuracy**
Detector1	0.636	0.757
Detector2	0.753	0.784
Detector3	0.764	0.848
Detector4	0.773	0.878
Detector5	0.815	0.905
Detector6	0.825	0.932
Detector7	0.859	0.956
Detector8	0.927	0.967
Detector9	0.949	0.983
Detector10	0.963	0.988

Fig 30. Two Different Training Network Parameters Setting on 5000 Testing Image

In the beginning, we set the training network that includes 3 convolutional layers, 32 feature filters with size 5x5, and 10 epochs to generate Detector1 through Detector10. We can see that the testing accuracy on 5000 testing images that increased as the number of training images increased. Detector1 was trained with 500 training images which reached an accuracy around 0.636, which is very poor. Detector10 which was trained with 5000 training images, the testing accuracy increased to 0.963. In contrast, we change training network to 5 convolutional layers, 64 feature filters with size 5x5 and 20 training epochs, and we generate new Detector1 through Detector10. This time, Detector1 was trained with 500 training images which reached an accuracy around 0.757, which is very better than the previous. Following the increase in the number of training images increased, Detector 1 through Detector 10 testing accuracy were improved. Detector10 which was trained with 5000 images, and its testing accuracy increased to 0.988. The testing accuracy difference of 0.025 indicates that the vehicle plate bounding box in each testing image could be correctly found with more confidence. When we changed training epoch to 30, the testing accuracy results became worse than the previous which was 0.951.

CHAPTER VI

DISCUSSION

The ten R-CNN training results show that the mini batch training accuracy has increased as the number of training epochs increased, and at the same time, the mini batch loss decreased. The training accuracy from Detector1 through Detector7 reached the highest point after the 6th epoch. However, Detector8 through Detector10 reached the highest point after 3th epoch. When trained different numbers of training images, the beginning points of accuracy and loss were different in each graph. For example, in 500 images training detector results, the accuracy started from 0.68, but some testing images could reach 0.988 accuracy. When training iteration reach 2 epochs, the accuracy values above 0.8 to 0.988 and the loss value started from 0.7 to 0.001. In 1000 images training detector results, the accuracy started from 0.37 and loss dropped from 2.5. Training detectors accuracy had increased dramatically during training and training accuracy reached 0.95 after the second epoch, then the training loss had dropped steeply and lower than 0.1 after the second epoch.

In R-CNN testing results, we see that using 5000 training images results in the most accurate detector, Detector10 reached the greatest testing accuracy after the 8th training epoch. For example, testing results, the attesting accuracy was 0.636. The testing accuracy moved higher from the 1000 images training detector to 5000 images training detector. In Detector10 training and testing results, the testing accuracy on 5000 testing images reached 0.963 initially, but after we modified the training network into 5 convolutional layers, 64 feature filters with size 5x5, and 20 training epochs, Detector10 reached the greatest training accuracy at the 7th epoch and the accuracy value was 0.988.

Using our 2nd network parameters to train a vehicle dataset that obtained from [67], the training dataset includes 4500 vehicle images and testing dataset include 4500 vehicle images. Each image resolution is 1920 X 1080 pixels.

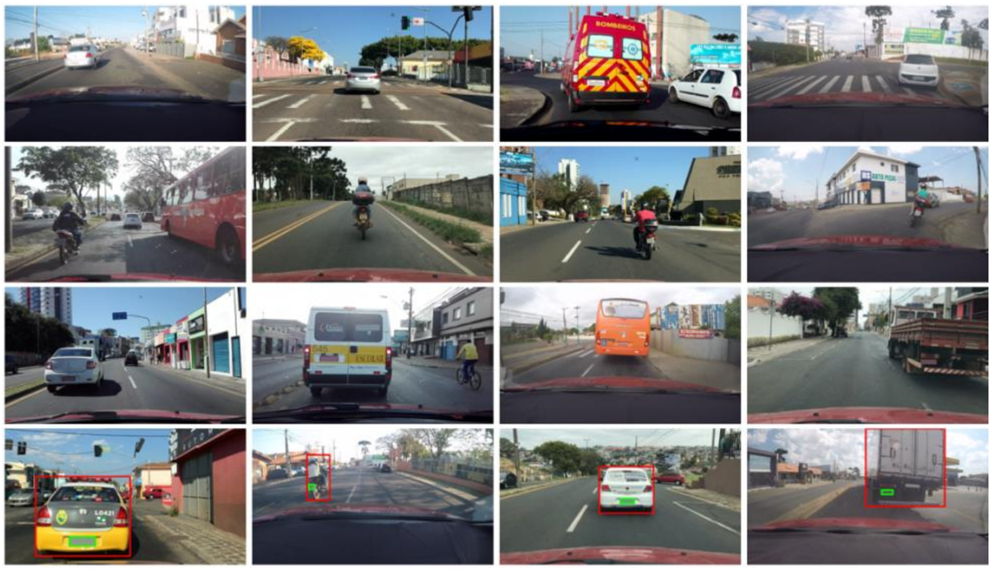


Fig 31. YoLo Training Dataset

RCNN Detector Accuracy	YoLo Detector Accuracy
94.5% on testing dataset	100% on testing dataset

In further experiment, we tried out the same training networks and parameters except the training epoch with 5000 training vehicle images. We increased the training epoch from 20 to 30. The training process shows that the training accuracy can reach the highest which is 0.99 after 20 epochs. After that, we use the trained detector to test on the 5000 testing images. The testing accuracy drops from the 20 epochs 0.98 to the 30 epochs 0.95, the training network should be overfitted.

CHAPTER VII

CONCLUSION

This project demonstrates that more training data benefits detector training errors and testing accuracy. Even for images with bad resolution, shadows and glints on vehicle surfaces, the detector found plates successfully.

Meanwhile, the R-CNN in this project could be sped up significantly by using the Faster-RCNN. The Faster-RCNN replaces the selective search algorithm (SSA) process with region proposal network (RPN). The Faster-RCNN could resize input images then put the anchor boxes to locate the correct detected object. It can be used in real time.

Although the R-CNN plate detection has good performance, this project is still unfinished. For example, the bounding box worked very well to detect the plate, but it still needs to be improved until it detects the plate edges more accurately with a standard size. Once this has been done, we can start to design a recognition system that uses blob extraction-based method [66] or a CNN method to classify characters and numbers from the plate, and to show the state the plate is from.

In the training process, we used Intel Core i7-4770 with four CPUs and one GeForce GT635 GPU machine. The parallel computing with four CPUs workers spends about 8 hour to train R-CNN. However, in the testing process, the trained R-CNN takes about 1.08 second to process one JPEG to detect the plate.

In Rafique’s method, they performed R-CNN within moving camera and moving vehicles, but they tested on less testing vehicle images which were about 1000. It took around 0.35s to detect car plate in a 640x480 pixel image. However, our machine tested on 5000 vehicle images and spent 1.08s to detect each car plate in a 2048x512 pixel image. This project was beginning with plate detection and there are more features that could be added like plate characters extraction and recognition.

References

[1] C. Bishop, *Pattern Recognition and Machine Learning*. Neural Networks., 2018.

[2] J. Brownlee, “A Gentle Introduction to Computer Vision,” *Machine Learning Mastery,* 2019.

[3] V. Bushaev, “Stochastic Gradient Descent with momentum,” *Medium, towards data science,* 2017.

[4] “Convolution Neural Networks for Visual Recognition,” *Machine Learning Mastery,* 2019.

[5] G. Cybenko, Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems, 2*(4), 303-314. doi:10.1007/bf02551274., 1989.

[6] V. S. Chandel, “Selective Search for Object Detection (C++ / Python),” *Learn OpenCV,*2017.

[7] MATLAB, Detect, 2019

[8] MATLAB, Detector, 2017

[9] B. Dickson, “What is computer vision?,” *PCMag,* 2019.

[10] M. T. Manry, “Neural Networks and Deep Learning,” *University of Texas at Arlington,* 2019.

[11] J. Frankenfield, “How Artificial Intelligence Works,” *innoplexus,* 2020.

[12] R. Gandhi, “R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms,” *towards data science,* 2018.

[13] P. Gavali, & J. Banu, “Deep Convolutional Neural Network for Image Classification on CUDA Platform,” SciencdDirect, 2019.

[14] R. Girshick, J. Donahue, T. Darrell, & J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2014.81, 2014.

[15] MATLAB, GroungTruth, 2019.

[16] “Image Segmentation in Deep Learning: Methods and Applications,” *missinglink.ai,* 2020.

[17] T. Karmarkar, “Region Proposal Network (RPN) - Backbone of Faster R-CNN,” *Medium,* 2018.

[18] R. Laroca, et al., “A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector,” *IEEE Xplore,* 2018 International Joint Conference on Neural Networks, Rio de Janeiro, 2018. pp. 1-10, doi: 10.1109/IJCNN.2018.8489629.

[19] H. Li, & C. Shen, “Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs,” arXiv:1601.05610 [cs.CV], 2016

[20] H. Li, P. Wang, & C. Shen, “Towards End-to-End Car License Plates Detection and Recognition with Deep Neural Networks,” arXiv:1709.08828 [cs.CV], 2017.

[21] P. Matthew Stewart, “Simple Introduction to Convolutional Neural Networks,” *Medium, towards date science,* 2020.

[22] “Multilayer Perceptron,” DeepLearning 0.1 documentation, Theano Development Team, 2013.

[23] E. Mihail, “Object Detection with R-CNN,” 2018.

[24] MATLAB, “ObjectDetectorTrainingData,” 2017.

[25] Pawangfg, “Selective Search for Object Detection: R-CNN,” *GeeksforGeeks,* 2020.

[26] A. Pinkus, “Approximation theory of the MLP model in neural networks,” *Acta Numerica, 8*, 143-195. doi:10.1017/s0962492900002919, 1999.

[27] J. C. Príncipe, N. R. Euliano, & W. C. Lefebvre, *Neural and adaptive systems: Fundamentals through simulations*. New York: Wiley, 1999.

[28] S. Pulkit, “Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques(Part1),” *Analytics Vidhya,* 2020.

[29] S. Pulkit, “A Step-by-Step Introduction to the Basic Object Detection Algorithms (Part1),” *Analytics Vidhya*, 2020.

[30] “Python Machine Learning Tutorial,” *Real Python,* 2020.

[31] M. A. Rafique, W. Pedrycz, & M. Jeon, “Vehicle license plate detection using region-based convolutional neural networks,” *Soft Computing, 22*(19), 6429-6440. doi:10.1007/s00500-017-2696-2, 2017.

[32] B. Rohrer, “How do Convolutional Neural Networks Work?,” *course.e2eML.school,* (2016).

[33] M. Rouse, “What is Artificial Intelligence (AI)?,” *TECHTARGET NETWORK*, 2020.

[34] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks‑the ELI5 way,” *Medium, towards data science*, 2018.

[35] “Support Vector Machines,” *scikit learn*, 2020.

[36] S. M. Silva, & C. R. Jung, “License Plate Detection and Recognition in Unconstrained Scenarios,” *Computer Vision – ECCV 2018 Lecture Notes in Computer Science,* 593-609. doi:10.1007/978-3-030-01258-8_36, 2018.

[37] P. Sharma, “A Practical Implementation of the Faster R-CNN Algorithm for Object Detection,” *Analytics Vidhya,* 2020.

[38] K. Spirina, & A. Zharovskikh, “How Does Computer Vision Work,” *InData Labs,* 2020.

[39] C. Thomas, “An introduction to Convolutional Neural Networks,” *Medium, towards data science,* 2019.

[40] MATLAB , “TrainingData,” 2019.

[41] MATLAB , “TrainFastRCNNObjectDetector,” 2019.

[42] Vision Online Marketing Team, “AI and Computer Vision Technology Diagnose Eye Diseases,” *Global Association for Vision Information,* 2019.

[43] L. Weng, “Object Detection for Dummies Part 3: R-CNN Family,” *Lil’Log*, 2017.

[44] R. Yamashita, M. Nishio, R. K. Do, & K. Togashi, Convolutional neural networks: An overview and application in radiology. *Insights into Imaging, 9*(4), 611-629. doi:10.1007/s13244-018-0639-9, 2018.

[45] A. Zharovskikh, & K. Spirina, “How Does Computer Vision Work,” *InData Labs,* 2020.

[46] D. Parthasarathy, “A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN,” *Athelas,* 2017.

[47] Techslang, “What is an Expert System in AI?,” *Techslang,* 2020.

[48] R. Gour, “What is Natural Language Processing in Artificial Intelligence?,” *Medium,* 2019.

[49] B. Marr, “What is Machine Vision And How Is It Used In Business Today?,” *Forbes,* 2019.

[50] AIsmartz, “Speech Recognition Using Artificial Intelligence,” 2019.

[51] A. Bonner, “The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks,” *Medium, towards data science,* 2019.

[52] P. Bhavsar, “Object Detection with Convolutional Neural Networks,” *Medium, Data Driven Investor,* 2019.

[53] S. H. Sumit, “Drawbacks of Convolutional Neural Networks,” *Sumit’Log,* 2018.

[54] L. Zou, et al., “License Plate Detection with Shallow and Deep CNNs in Complex Environments,” *Complexity, 2018*, 1-6. doi:10.1155/2018/7984653, 2018.

[55] J. Ian, et al., “Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks,” *arXin:1312.6082 [cs.CV],*2014.

[56] G. E. Hinton, et al., “Improving neural networks by preventing co-adaptation of feature detectors,” *Technical report, arXiv:1207.0580*, 2012.

[57] A. Hyvarinen, J. Karhunen, & E. Oja, “Independent Component Analysis,” *Wiley-Interscience,* 2012.

[58] P. Sermanet, S. Chintala, & Y. LeCun, “Convolutional neural networks applied to house numbers digital classification,” in *international Conference on Pattern Recognition* (ICPR 2012), 2012.

[59] S. Ananth, “R-CNN for Object detection,” 2019.

[60] A. Rosebroke, “Intersection over Union (IoU) for object detection,” *pyimagesearch,* 2016.

[61] D. Tikk, L. T. Kóczy, & T. D. Gedeon, “A survey on universal approximation and its limits in soft computing techniques,” *International Journal of Approximate Reasoning, 33*(2), 185-202. doi:10.1016/s0888-613x(03)00021-5, 2003.

[62] V. S. Chandel, “Selective Search for Object Detection (C++/Python),” *Learn OpenCV*, 2017.

[63] S. Gaurav, “Deep Learninng method for object detection:R-CNN explained,” *towards data science*, 2020.

[64] C.L Zitnick, P. Dollar, “Edge Boxes: Location Object Proposals from Edges,” *Microsoft Research*, 2014.

[65] S. McMahon, N. Sunderhauf, B. Upcroft & M. Milford, “How Good Are Edge Boxes, Really?,”*Australian Center for Robotic Vision*(ACRV).

[66] V. Karthikeyan, V. J. Vijayalakshmi, P. Jeyakumar, “License Plate Segmentation Based on Connected Component Analysis,” *ECE Coimbatore, India.*

[67] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, D. Menotti, “*A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector*,” in 2018 International Joint Conference on Neural Networks (IJCNN), July 2018, pp. 1–10.