License Plate Detection Using One-stage Object Detection Algorithms

Niloofar Baghdadi

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Science (Computer Science) at
Concordia University
Montréal, Québec, Canada

November 2020

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Niloofar Baghdadi**

Entitled: **License Plate Detection Using One-stage Object Detection Algorithms**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

______________________________ Chair
Dr. Tse-Hsun Chen

______________________________ Examiner
Dr. Adam Krzyzak

______________________________ Examiner
Dr. Tse-Hsun Chen

______________________________ Supervisor
Dr. Ching Y. Suen

Approved by ______________________________
Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

November 30, 2020 ______________________________
Mourad Debbabi, Interim Dean
Faculty of Engineering and Computer Science

Abstract

License Plate Detection Using One-stage Object Detection Algorithms

Niloofar Baghdadi

Automatic License Plate Detection and Recognition (ALPR) has many practical applications such as traffic control and parking tickets; for this reason, it has been one of the exciting research topics. Environmental factors such as lighting and dust, make automatic license plate detection and recognition challenging, especially for traditional image processing methods. Although much research has been conducted on ALPR systems using image processing and computer vision tools and algorithms, the need for more research on this topic with deep-learning algorithms has not been satisfied yet. Among different and in succession phases of ALPR, the license plate detection phase is of great importance because it is the first phase, and its performance affects the result of other stages. Moreover, due to the advent of technology and artificial intelligence in everyday life, having reliable real-time ALPR systems is necessary. Hence, this work empirically studies the mean Average Precision (mAP) of Single Shot MultiBox Detector (SSD) and You Only Look Once (YOLOv4) on CENPARMI and UFPR-ALPR datasets. Although we achieved good mAP results of 95.47 % (ResNet-SSD) and 95.45 % (InceptionV2-SSD) with the SSD model during this experiment, we have reached the highest mAP of 97.46 % and 97.78 % with the newly released YOLOv4 model on CENPARMI and UFPR-ALPR datasets, respectively. However, in object detection, high precision is not the only essential criterion anymore. Hence, we scrutinized the object-detectors mentioned above to find a model that can balance mAP, speed, and memory. We learned that the higher the number of parameters of a model, the better the detection results. On the other hand, the number of parameters of a model can affect an object detection task's speed.

Acknowledgments

I want to give my deepest and sincerest gratitude to those who have helped me in this endeavor. I will never forget my supervisor Dr. Ching Y. Suen, whom his guidance, patience, understanding, and kindness helped me through all the challenges. I was truly fortunate to work under his supervision and be rewarded with lifelong lessons. I remain thankful to my family and friends, who gave me the courage and support to continue to walk toward my dreams. I am also forever grateful to anyone who had taught me anything throughout my journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

License Plate Detection and Recognition is essential for many purposes, such as traffic control, finding stolen vehicles, issuing parking tickets, and so forth. There is quite a few research studies in license plate detection and recognition; however, there is always a need for improved and robust techniques to replace the old ones. In the field of license plate detection and recognition, we are dealing with still images or videos of vehicles passing by.

To have an Automatic License Plate Detection and Recognition (ALPR), one should accomplish different phases. License plate detection, character segmentation, and recognition are among the main phases that are in succession. Hence, the performance in one phase affects the result of the other. If we acquire good results in the early stages, having a reliable ALPR system can be promising. Besides, every jurisdiction uses different background colors, graphics, and fonts in license plates to make them unique and recognizable; however, this can affect the ALPR systems' recognition phase [3].

The use of artificial intelligence, especially deep learning, in science and technology, is becoming prevailing. Many researchers studied license plate detection and recognition systems using image processing and computer vision tools and algorithms, while more research is required using brand-new deep learning algorithms. To this extent, this study is devoted to the detection of license plates using very recent object detection algorithms in the area of deep learning.

Since the ALPR systems are more applicable in real-time environments, we concluded work on one-stage detection models that are remarkably faster than two-stage detection models. After a careful review of research conducted by scholars about license plate detection with Single Shot MultiBox Detector (SSD) algorithm [4][5][6], variants of YouOnly Look Once (YOLO) algorithm [7][8][9][10][11], and Vanilla Convolutional Neural Network(CNN) [12][13][14][15][16][17][18][19], we noticed there is plenty of research done with the CNN and YOLO models. In contrast, we found less research on the SSD model for license plate detection. Therefore, we examined two of the one-stage object detection algorithms, the SSD and YOLOv4, with two different datasets during this study. While in this research, the emphasis is more on the SSD algorithm by scrutinizing different feature extractors.

The challenge with deep learning algorithms is that much data is required to have a valid prediction. In this study, we used two different datasets to train our models. Except for the Center for Pattern Recognition and Machine Intelligence (CENPARMI), the Federal University of Parana dataset for Automatic License Plate Recognition (UFPR-ALPR)[1] is publicly available upon request for non-business purposes. The CENPARMI dataset consists of license plates' images from the United States and Canada, while the UFPR-ALPR dataset consists of license plates' images from Brazil.

We learned that the higher the number of parameters of a model, the better the detection results. On the other hand, the number of parameters of a model can affect an object detection task's speed. Although we trained the selected models on pre-trained networks, we presume more data can help the model to generalize better and provide better results.

In the end, we reached promising results from both SSD and YOLOv4 object detection models on the aforementioned datasets. However, depending on an ALPR system's application, one should try to balance and choose between speed, accuracy, and memory. If the speed and accuracy are

[1] `https://web.inf.ufpr.br/vri/databases/ufpr-alpr/`

essential, YOLOv4 can be an option for the detection phase, and if speed and memory are essential SSD model can be a better option.

Following is the structure of this project: We review some related works regarding ALPR systems in chapter 2. Chapter 3 encompasses details of different models that we employ in our project. Then, we discuss and report the results we acquired in this project in chapter 4. Eventually, in chapter 5, we conclude our work and propose some ideas for future works.

Chapter 2

Literature Review

2.1 License Plate Transformation

A license plate or vehicle registration plate is a metallic plate used to identify a motor vehicle. As is shown in Figure 2.1, North America's plate size is 12×6 inches and narrower than most license plates used in Europe with 20.5×4.5 inches. The creation and evolution of license plates alongside the issues they have in legibility and readability are areas in which researchers should dive in more. In the following, we describe the history of the license plate on a short scale.

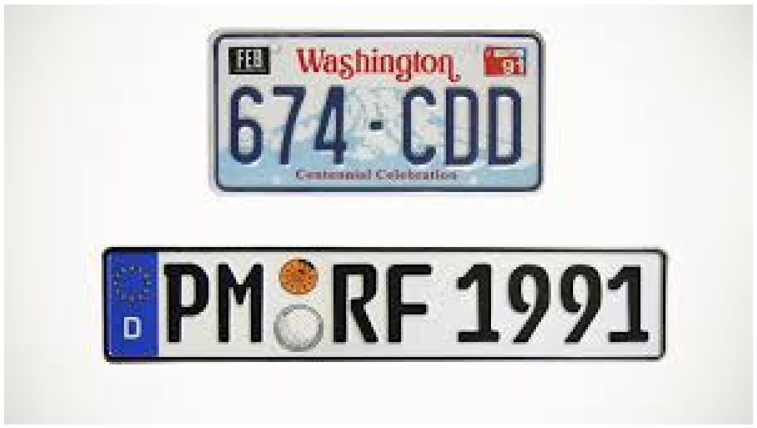


Figure 2.1: Sample license plate in American vs. European countries.

Figure from: https://www.autoweek.com/car-life/a2138056/autoweek-asks-should-european-format-plates-be-option-america/

- **Issuance**

Many changes have taken place related to the issuance of vehicles' license plates in terms of fonts, background color, graphic, and base materials. The very first license plate had been issued in Philadelphia, Pennsylvania, in the 1850s. [3].

Later in 1901, the state of New York required vehicles to have license plates, and in 1903 a standard license plate was issued through Massachusetts. In the long run, in 1915, most of the states required license plates for vehicles and charged the owners for a registration fee every year as a source of transportation revenue [3].

- **Retro-Reflective Technology**

As stated by [3], the growth of the population, the number of motor vehicles increased. Hence, other than vehicle identification, vehicle safety became important. Hence, retro-reflective license plates were issued by New Mexico in 1936 to increase vehicles' visibility at night.

Eventually, all of the U.S. states, two provinces of Newfoundland and Alberta in Canada, and countries like San Marino, Peru, and Costa Rica made use of the retro-reflective technology, which reduced the number of accidents at night by improving the legibility of license plates of vehicles [3].

- **Material**

According to [3], steel was the primary material used to make license plates in the early 20th century; however, due to the second world war and the high steel demand, States jurisdictions issued only one license plate for each vehicle.

Nevertheless, after the second world war, some states switched to using license plates for the front and rear in motor vehicles, while others kept the tradition of having only one license plate to save costs. It is worth mentioning that nowadays, aluminum is the primary material used in making license plates of vehicles [3].

- **Background Color, Graphics, and Fonts**

Different jurisdictions use different colors for the background and the alphanumeric characters to recognize their plates easier. By increasing the number of vehicles, graphics were added to the license plates to make them distinct and recognizable by the State's jurisdictions. However, nowadays, by a myriad of graphics, background and alphanumeric colors, and fonts, the license plates' legibility is becoming problematic.

As stated by the authors in [20], there is no official font for license plates used by North American countries despite European countries and elsewhere. Although the fonts that jurisdiction uses for the license plates in North America look consistent, the necessity of having a standard font in Northern American countries will be of great help in legibility and readability of license plates.

In agreement with [3], "the lack of national standards regarding the design and manufacturing of license plates, limits the effectiveness of ALPR technology."

2.2 License Plate Detection and Recognition

Although consistent changes to the manufacturing by the advent of technology help to have more readable and legible license plates, the Automatic License Plate Detection and Recognition (ALPR) System will be another solution to improve the readability of the license plates of vehicles. As stated by [3], the main stages of constructing an ALPR system are as follows:

- Detecting and locating the license plate
- Extracting the characters from the license plate
- Identifying the license plate and jurisdiction

It should be noted that most of the research has been conducted in the area of ALPR is on the first two stages mentioned above. ALPR system has different applications such as detecting stolen cars by law enforcement, traffic and border control, and parking security, for instance.

It is always easier to have a text on the computer screen and transform it into a physical piece of paper than vice versa. Optical Character Recognition (OCR) is a technique that converts images of handwritten or machine-printed text into machine-readable characters. This technique can be utilized in license plate detection and recognition as well. Many scholars conducted exciting research on implementing an ALPR system with the help of OCR technology.

Therefore, it is worth evaluating distinct ways of detecting and recognizing vehicles' license plates and comparing them to help organizations using this application. There are numerous ways to implement an ALPR system; however, we only review some of the research studies conducted in the Neural Network (NN) and Template Matching methods of Optical Character Recognition (OCR) in this study.

2.2.1 ALPR with Template Matching

The authors in [21] proposed a technique for the detection and recognition of Indian license plates. They used the template matching technique, which recognizes the characters inside an image by comparing them with the given templates. The authors integrated the English and Hindi characters in their work, and before performing the template matching method, they performed the morphological and threshold operations on images as their proposed method. If the characters do not exist in any of the templates, they described it as an invalid license plate. The authors achieved the localization, segmentation, and recognition rate of 92 %, 97 %, and 98 %.

In another work, authors in [22] conducted an empirical study on license plate detection and recognition with three approaches. The authors used OCR with template matching,multi-class support vector machine (SVM), and convolutional neural network (CNN) in their study. They compared each of the approaches' performance and noticed that the CNN approach performs better than the other two approaches. This study achieved an accuracy of 88.23 %, 92.64 %, and97.06 % for template matching, multi-class support vector machine, and convolutional neural network, respectively.

Besides, authors in [22] stated that CNN's perform better because they can learn from any license plates with different fonts and designs. Consequently, it can be valuable and of great importance to scrutinize license plate detection and recognition tasks in the novel neural Network way.

2.2.2 ALPR with Neural Networks

Different scholars researched Automatic License Plate Detection and recognition based on image processing, computer vision algorithms, and Optical Character Recognition (OCR) technology. However, according to [23], there is a great deal of information in an image that deep Learning algorithms consider them for visual recognition while computer vision algorithms do not.

Moreover, by the increasing popularity of Machine Learning and Deep Learning fields, many exciting research studies about ALPR are being conducted in those fields. Although in demand, Deep Learning algorithms need massive datasets to perform reasonably well. Nonetheless, an abundance of exceptional work in the Deep Learning field has been done to either syntactically expand the data or create powerful ways to overcome the issues.

Based on the stages mentioned above in implementing an ALPR system, some scholars conducted research only in the first phase, the detection phase, and some others worked on both detection and recognition phases. All the phases are critical, yet satisfactory detection can lead the way to a robust ALPR system.

- **Detection based on Single Shot MultiBox Detector(SSD)**

The work in [4] presented implementing the Single Shot Detection (SSD) algorithm for license plate detection. The authors stated that by replacing the base algorithm of SSD, the VGG Network, with the Residual Network (ResNet), they could better detect license plates. Hence, they use ResNet due to its characteristic that does not allow the gradient to vanish despite being a deep network. The authors achieved an 85.5 % average accuracy score with ResNet, whereas they achieved an average accuracy of 83.6 % with the VGG network.

The authors in [5] studied the license plate localization phase of an ALPR system and compared four different algorithms on a new dataset. The dataset is labeled for both cars and license plates. The author compared the SSD algorithm once with MobileNet and another time with Resnet50 feature extractors, Faster R-CNN algorithm with the Inception base, and R-FCN algorithm with Resnet101 base. They observed that SSD's accuracy with MobileNet was higher than the others for car detection and the accuracy of FasterR-CNN with Inception was higher for License Plate detection. The author achieved an accuracy of 98 % on SSD and 90 % on Faster R-CNN.

In general, the Single Shot Multibox Detector (SSD) is a small and easy algorithm to train that can be used for real-time applications. The authors in [6] used the SSD algorithm with the MobileNet feature extractor for localizing the license plates, and on top of the SSD algorithm, they used the Convolutional Neural Network (CNN) for character recognition. The authors aimed to show that new deep-learning algorithms tend to have better results than previous traditional methods. They evaluated the final model with different test sets, called standard and side view sets. They achieved the detection accuracy of 92 % on the standard set and 85 % on the side view set.

Although researchers have done numerous research in License plate detection with deep learning algorithms, some different scenarios and algorithms are still worth exploring for the detection phase of an ALPR system.

- **Detection based on You Only Look Once (YOLO)**

The authors in [7] proposed Multi-Directional Car Plate Detection, using You Only Look Once (YOLO) architecture. The authors performed some refinements on the YOLO object detection architecture to make it work in situations that cameras have different degrees of rotation while taking videos or pictures. They call the model mentioned above, Multi-Directional YOLO (MD-YOLO). The authors applied the attention method to estimate the precise location of the license plates. Afterward, the estimated region is passed to the MD-YOLO. The authors achieved acceptable results on different subsets of the AOLP[1] dataset. They achieved the precision of 99.51 %, 99.43 %, and 99.46 % on Access Control, Traffic Law Enforcement, and Road Patrol subsets of AOLP, respectively.

The authors in [8] applied the YOLO object detector on the SSIG dataset for detection, and for the recognition, they used three CNN networks, one for character segmentation and two others for

[1]Application Oriented License Plate (AOLP): `http://aolpr.ntust.edu.tw/lab/`

digit and letter recognition. They achieved a better result than two commercial systems, such as Sighthound and OpenALPR. Then the authors introduced a public dataset named UFPR-ALPR. They achieved the recognition rate of 78.33 % on the new dataset with their system, while the commercial systems achieved below 70 % recognition rate.

In [9], the authors refined the FAST-YOLO network to extract both frontal views of the car and the license plate. They adjusted the Fast-YOLO to output both of car and license plate for the detection phase. For character detection and recognition, the authors modified YOLO architecture and applied a heuristic approach to improving their final result. The authors stated that they achieved 63.18 % accuracy, while the Sighthound achieved 55.47 % for correctly detected and recognized license plates.

The authors in [10] represented their work by implementing an ALPR system using three YOLO networks for car detection, license plate detection, and character detection, respectively. In the end, the authors used a CNN network for character recognition. They made their dataset of 604 car images and tested their work on them. The authors stated that all the stages' overall validation accuracy exceeded 90 %; however, the system accuracy on 50 test images reached only 82 % with some fault tolerance.

In [11], the authors researched on multilingual license plate detection and recognition on two datasets. They proposed an end-to-end license plate detection and recognition system. For the detection phase, the authors used YOLOv2 and for the recognition phase; however, in the recognition phase, they examined segmentation-free (i.e., Recurrent Neural Network)and segmentation-based (i.e., YOLOv2) algorithms. The authors compared the result of the recognition phase and noticed that segmentation-based algorithms perform better for multilingual license plates. They achieved a recall rate of 99.09 % and 100 % on Radar and GAP-LP datasets for the IoU threshold of 0.5, respectively. Moreover, the authors achieved 91.46 % and 95 % recognition rate with the YOLO algorithm and recognition rate of 42.88 % and 95.88 % with the RNN algorithm on Radar and GAP-LP datasets.

- **Detection based on Convolutional Neural Network (CNN)**

Not all the researchers put an end to their exploration after the license plate detection phase. A great deal of valuable work has been done in the ALPR system's recognition phase, similarly. The work in [12] implemented a Convolutional Neural Network (CNN) for feature extraction, and the result of the CNN is passed to a Recurrent Neural Network (RNN) with36 hidden units for sequencing the characters of license plates. They achieved 76 % accuracy in recognizing the characters in license plates and 95.1 % accuracy per character.

The authors in [13] trained a 37-classes CNN to detect license plates, and in order to remove the false positives, they trained another two classes of CNN. In the end, they trained an RNN for character recognition with a precision of 97.56 % and a recall of 95.24 %. Another time, Li and Shen, with the company of Wang, tackled the Automatic License Plate Detection and Recognition. They proposed a single network mainly containing layers of CNN and RNN alongside other required layers to design an end-to-end ALPR system. They showed that they improved the performance[2] of their previous work (two-stage CNN-RNN network with the performance of 94.09 %) and achieved the performance of 97.13 % [14].

The authors in [15] used the Generative Adversarial Network (GAN) to improve their system accuracy by generating images. They trained a CNN and RNN with images generated byGAN to create a pre-trained network and fine-tuned the pre-trained network on their dataset. They were able to achieve 92.1 % of recognition accuracy and 98 % of character recognition accuracy.

The authors in [16] proposed an end-to-end commercial ALPR system named Sighthound with three CNN networks for license plate detection, character detection, and recognition. The first CNN used to localize the license Plates, the second CNN, which has two classes used for detecting plates versus non-plate images, and the third CNN, which has 35 classes used for recognizing the

[2]Measured by Intersection over Union (IoU)

characters inside the license plates. The authors achieved the recall[3] of 99.09 % on the US license plates and 99.64 % on European license plates.

In [17], the authors performed some preprocessing with edge-based methods to find the potential regions for license plates, then two classes CNN is used to remove the false positives. Later on, the authors used 11 parallel CNN (the reason was that the license plates they were dealing with had 11 alphanumeric characters) with 37 classes for character recognition. The authors were able to accomplish license plate recognition accuracy of 97 % for single line license plates.

Moreover, the authors in [18] used some image processing methods as a preprocessing step to localize the region of interest in images. Afterward, they used 2-class CNN to distinguish plates and non-plate regions. Segmentation of characters in detected license plates from the previous task is another step of their work. The authors used image processing methods to do character segmentation before character recognition with a 37-class CNN. In the end, the authors were able to achieve an accuracy rate of 94.8 % on the Caltech dataset and the accuracy rate of 96.2 %, 95.4 %, and 95.1 % on Access Control, Traffic Law Enforcement, and Road Patrol subsets of the AOLP dataset, respectively.

The work in [19] presented a new baseline named The Roadside Parking Net (RPnet)for Automatic License Plate Detection and Recognition on a large dataset named CCPD[4]. The authors introduced the RPnet as a single network which is composed of the detection and recognition modules. In the first module, a ten layer CNN creates feature maps and passes the feature maps to 3 fully connected layers to predict boxes. In the second module, there are some ROI pooling layers and classifiers to predict the license plate alphanumeric characters. The authors illustrated that their model achieved better results in terms of speed and accuracy than other state-of-the-art approaches on a similar dataset. The authors achieved the detection precision of 94.5 % and the recognition accuracy of 95.5 %.

[3]Recall is a fraction of true positive over predicted results

[4]Chinese City Parking Dataset

Last but not least, License plate detection and recognition has been studied from different perspectives. Nevertheless, their improvement hinges on the advancement of technology and demands. Hence, as the technology evolves, one would want to explore more on the topic.

Chapter 3

Methodology

3.1 Object Recognition Overview

Object Recognition is a term that encompasses other computer vision tasks; however, it mostly refers to "object detection" when it is used [24]. Object recognition includes but is not limited to:

- **Image Classification**: Image classification is a task which predicts the class of an object in the image. The mean classification error assesses an image classification model's performance over the predicted class labels.
- **Object Localization**: Object localization finds the object with its bounding box information. An object localization model's performance is assessed by the distance between the expected and predicted bounding box.
- **Object Detection**: Object detection proposes the class of the object and its location with a bounding box. The precision and recall assess an object detection model's performance over each of the best matching bounding boxes for the image's known object.

To better understand how object detection in the field of neural networks works, one should dive more into the Convolutional Neural Networks(CNNs) that pave the way for the recent progress in image classification hence object detection.

3.1.1 Convolutional Neural Networks (CNNs/ConvNets)

Convolutional Neural Network [25] is a neural network that implements a mathematical operation called convolution in at least one of their layers [26]. The convolution operation is written as follows:

$$s(t) = \int x(a)w(t-a)da \tag{1}$$

In the above equation, **x** is the input, **w** is the kernel, and **s(t)** is the feature map or output. Since we work with data on computers, the time is discrete, and the convolution we are interested in is the discrete convolution [26]. Moreover, as we work with 2-D image **I**, the kernel **K** should be 2-D consequently.
Listed below is the discrete convolution formula for 2-D images:

$$S(i,j) = (I*K)(i,j) = \Sigma_m \Sigma_n I(m,n)K(i-m, j-n). \tag{2}$$

Convolutional Neural Network or ConvNet consists of three main layers, Convolutional Layer, Pooling Layer, and Fully-Connected Layer. Figure 3.1 shows the architecture of CNN.

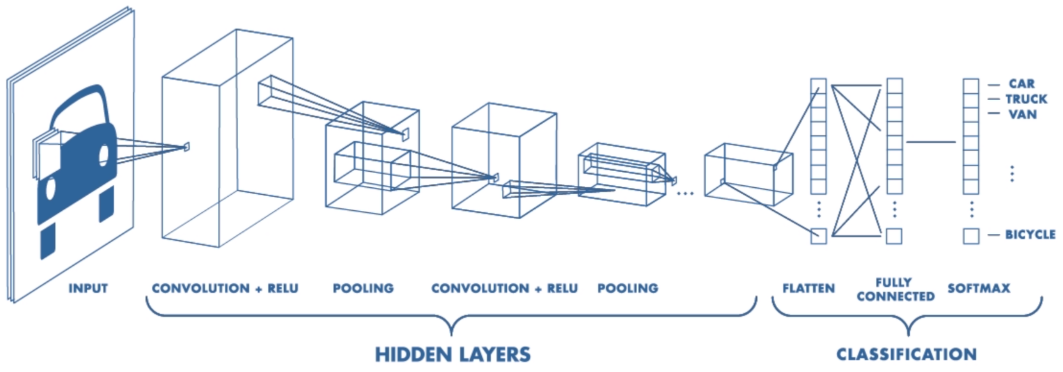


Figure 3.1: Architecture of a CNN.

Figure from: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks–1489512765771.html

Convolutional Layer is the core building block of a convolutional network and performs convolution operation spatially on the input volume with a set of learnable filters[1]. As it is clear from Figure 3.2, each filter produces a 2-dimensional activation map, and the output volume will be produced by stacking the activation maps.

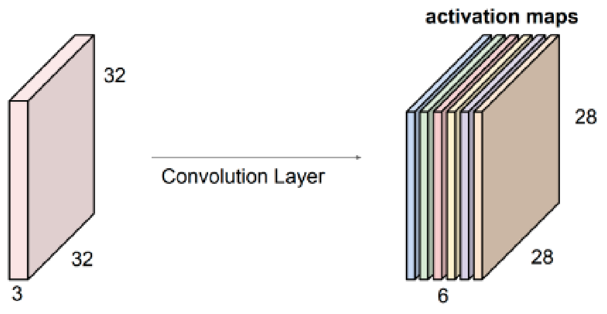


Figure 3.2: Effect of Convolutional Layer with a $5 \times 5 \times 3$ filter.

Figure from [27]

[1]Denoted as $\mathbf{K}$ in the equation (2)

A non-linear activation function always follows a convolutional layer in ConvNets. Activation functions have a significant effect on the output of neural networks. Rectified LinearUnit Layer (RELU) is an activation function mostly used in many types of neural networks because it is proven that a model that uses this function is more comfortable to train and often achieves better results [28]. Figure 3.3 shows popular activation functions.

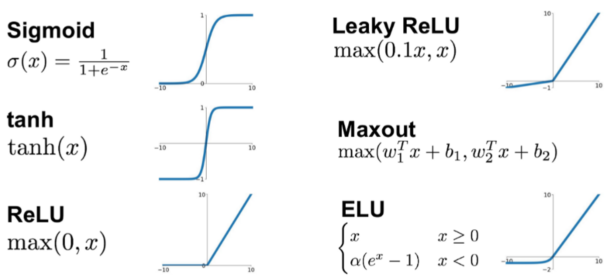


Figure 3.3: Different activation functions and their associated graphs.

Figure from: https://mc.ai/complete-guide-of-activation-functions/

The pooling layer is another layer in the convolutional network, which downsamples the feature map (or activation map) to make it invariant to small translations of the input image, reduce the number of parameters and computation of the network. The most popular downsampling operation is max pooling, as shown in Figure 3.4, in which a 2×2 filter, usually with a stride of 2 along width and height, will downsample the activation maps.

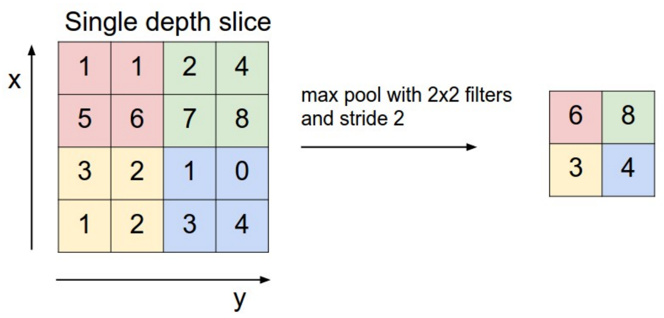


Figure 3.4: Max pooling operation.

Figure from [27]

Despite convolutional layers, neurons in a *Fully-Connected Layer* have full connection to the feature map in the previous layer. The fully-connected layer aims to classify the input image into different classes when convolutional, and pooling layers have performed feature extraction. By stacking the layers mentioned above, we can make a ConvNet architecture. Figure 3.5 shows a sequence of layers in a ConvNet that transform one volume of activation to another.

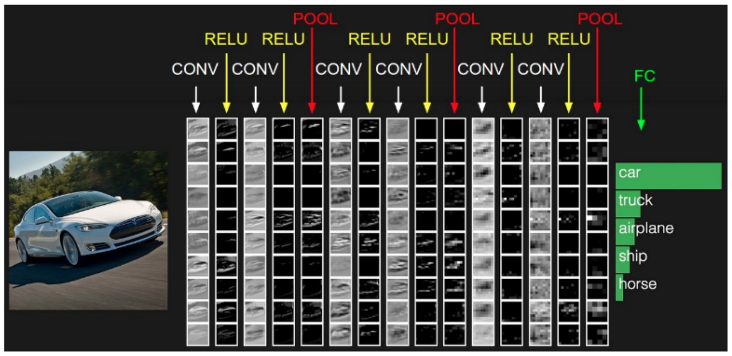


Figure 3.5: The activations of an example ConvNet architecture.

Figure from [27]

Being well acquainted with ConvNets, many scientists have developed and are developing different object detection algorithms by enriching the ConvNets as feature extractors with different modules, all of which have their pros and cons over the other. Hence, depending on the application, selecting one object detector might be advantageous to the other. Nevertheless, one should consider that accuracy is not the only criterion of interest or importance in object detection applications. Memory and speed are another crucial basis which would affect the selection of a proper object detector.

For instance, fast detection algorithms are more efficient if the application is similar to autonomous driving, which needs the objects to be detected faster. Some other applications require higher accuracy than speed, so in that case, accurate detection models are of interest.

3.1.2 Accurate Detection Models

Region-Based Convolutional Neural Network (R-CNN) family consists of R-CNN [29], FastR-CNN [30], Faster R-CNN [31], and Mask R-CNN [32], each of which can be used for object recognition. R-CNN, the first algorithm in this family, was introduced in 2014. Afterward, the Fast R-CNN was introduced in 2015 and improved the R-CNN. Likewise, the Faster R-CNN presented in 2017 boosted the Fast R-CNN algorithm, and lastly, the mask R-CNN algorithm was introduced in 2017 and extended the Faster R-CNN algorithm to pixel-level image segmentation. Although object recognition with the R-CNN family is accurate, these algorithms are slow because of the region proposal step involved in each of the algorithms mentioned above.

3.1.3 Fast Detection Models

Notwithstanding the R-CNN family that the object detection happens in two stages[2], the Single Shot Multi-box Detector (SSD) [33] and You Only Look Once (YOLO) [34] family are one-stage object recognition algorithms, which led them to be known as fast detection algorithms and hence real-time object detectors.

3.2 Object Detection Measuring Metric

Object detection metrics that are being used, established throughout the object detection challenges such as the PASCAL VOC Challenge[3], the COCO Object Detection Challenge[4], the Open Images Challenge[5], and the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[6].

[2]Region or box proposal stage which is done by the Selective Search algorithm or Region Proposal Network (RPN), and the classification stage.

[3]http://host.robots.ox.ac.uk/pascal/VOC/voc2012/htmldoc/devkit_doc.html

[4]http://cocodataset.org/#detection-eval

[5]https://storage.googleapis.com/openimages/web/object_detection_metric.html

[6]http://www.image-net.org/challenges/LSVRC/

The **mean Average Precision (mAP)** is an essential metric for the object detection tasks; however, some of the challenges, as mentioned earlier, like the COCO Object Detection Challenge, also include other metrics such as mean Average Recall(mAR) for evaluating object detection. To better understand the mAP, it is better to first dive more into the concept of True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN), Precision, Recall, and Intersection over Union (IoU).

The following are the necessary components of other metrics used in the object detection task, so in order to better grasp other metrics, one should digest the concept of each of the items concerning the object detection. According to the author in [35], the threshold can be set to 50 %, 75 %, or 95 %, depending on the metric.

- **True Positive (TP)**: A correct detection. Detection with IOU $\geq$ threshold
- **False Positive (FP)**: A false detection. Detection with IOU $<$ threshold
- **False Negative (FN)**: A ground truth not detected
- **True Negative (TN)**: Does not apply to an object detection task.

Having the metrics mentioned above in mind, one can better understand what precision and recall could mean in the object detection task.

Precision is depicted in Figure 3.6 and shows how accurate the prediction is. As is illustrated in equation(3), precision is the number of true positives divided by the sum of true positives(TP) and false positives(FP).

$$Precision = \frac{TP}{TP+FP} = \frac{True\ Object\ Detection}{All\ Detections} \tag{3}$$

Recall is depicted in Figure 3.6 and shows how good the model finds positive cases. As is shown in equation(4), recall is defined as the number of true positives(TP) divided by the sum of true positives(TP) and false negatives(FN).

$$Recall = \frac{TP}{TP + FN} = \frac{True\ Object\ Detection}{All\ Groundtruth} \tag{4}$$

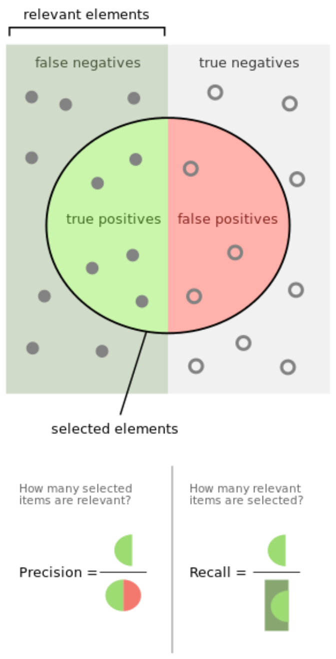


Figure 3.6: Precision vs. Recall

Figure from: https://en.wikipedia.org/wiki/Precision_and_recall

The **Intersection over Union (IoU)** computes the overlap between the predicted bounding box and the ground truth bounding. A predefined threshold determines whether the predicted bounding box is acceptable or not. Figure 3.7 is the graphical representation of the IoU.

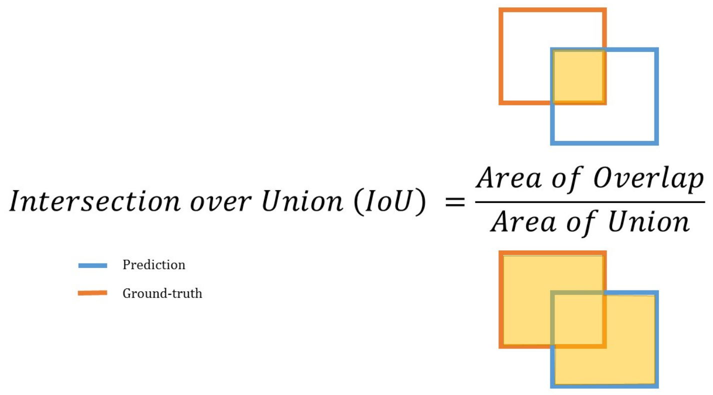

Intersection over Union (IoU) = Area of Overlap / Area of Union
Prediction
Ground-truth

Figure 3.7: Intersection over Union

Figure from: https://www.kaggle.com/c/3d-object-detection-for-autonomous-vehicles/overview/evaluation

Being well-acquainted with the evaluation metrics mentioned above, we can better grasp the concept of other evaluation metrics such as average precision(AP) and thence the mean average precision (mAP).

A precision-recall curve can also be used for evaluating the performance of an object detector; however, having a numerical value is better when comparing the results of an object detector. Hence, the average precision is being used, which is the area under the precision-recall curve.

The AP is shown mathematically in the equation (5), where P_n and R_n are the precision and recall at the n^{th} threshold [36].

$$\mathrm{AP} = \sum_{n} (R_n - R_{n-1}) P_n \tag{5}$$

The mean average precision(mAP) is the mean of average precision over all distinct C classes in the object detection task and is shown in equation (6).

$$\mathrm{mAP} = \frac{\sum_{i=1}^{C} AP_i}{C} \tag{6}$$

To have a better understanding of how well different object detectors are performing, it is better to have a unique evaluation metric as is provided by the authors of object detection algorithms.

3.3 The Choice of Algorithm

License Plate Detection and Recognition (LPDR) is a real-life and real-time application. For this reason, we have decided to evaluate a model to see how we can balance out the speed and accuracy. Although the selection of the algorithm for speed and accuracy trade-off is essential, other critical components exist that should be considered, such as the dataset, the type of feature extractor, the deep learning software platform, training configuration, and so on forth.

Moreover, we cannot compare different object detection algorithms because even though the authors' announced results are on identical datasets such as COCO[7] and PASCAL VOC[8], they

[7]http://cocodataset.org/#home
[8]http://host.robots.ox.ac.uk/pascal/VOC/

performed the experiments in different settings. Hence, depending on the application, one should select an object detector to a trade-off between accuracy and speed.

For Automatic License Plate detection and recognition(ALPR), we decided to examine the SSD model, one of the single-stage networks, and a fast detection model. Another reason behind choosing this network is that the speed is crucial in ALPR, so when we combine SSD with a feature extractor with fewer parameters, it can compete with other fast and accurate detection algorithms. This characteristic helps to avoid over-fitting during training the model when there is an insufficient amount of data.

As mentioned earlier, we cannot compare different object detection algorithms with different base feature extractors[9] in different settings. However, an extensive survey [37] has been done by the Google Research team in a more controlled environment about the trade-off between accuracy, memory, and speed of some object detection algorithms such as the Faster R-CNN, R-FCN [38], and SSD. The authors designed a unified platform named Object Detection API[10], in which the architecture of the Faster R-CNN, R-FCN, and SSD object detection algorithms are available in the TensorFlow Deep Learning library. As we examine the SSD model for the ALPR project, a more detailed explanation about the SSD model is provided in the following.

3.4 Single Shot Multi-box Detector(SSD)

Although two-stage detection models are known to be accurate, they are not computationally suitable for embedded systems and not fast enough for real-time applications. As a one-stage detector, the SSD has a higher speed in the detection and can compete with other accurate detection models.

[9]e.g., VGG, Mobilenet, Residual Networks
[10]https://github.com/tensorflow/models/tree/master/research/object_detection

By eliminating the region or box proposal stage of the two-stage object detectors, we can have a single-stage model that is simple, fast, easy to train, and straightforward to integrate into systems. SSD300[11] achieves 74.3 % mAP on VOC2007 test at 59 FPS[12] and SSD500[13] achieves 76.9 % mAP [33].

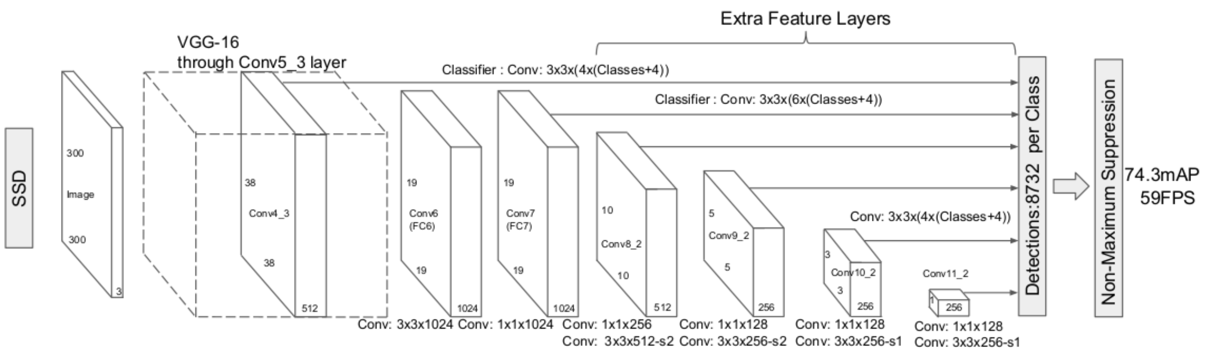


Figure 3.8: Single Shot Multi-Box Object Detector Architecture

Figure from [33]

During the feed-forward process, the SSD produces a fixed-size group of bounding boxes and object class instance scores in those boxes. The original SSD model uses the VGG-16 [39] model pre-trained on ImageNet as the base model; however, substituting networks such as MobileNet and Inception is also possible. On top of the VGG-16 (truncated before classification), there are convolutional feature layers of decreasing sizes that allow the prediction of detections at multiple scales. There is a non-maximum suppression step in the final layer to produce the final detections and remove the boxes whose score falls below a certain threshold [33]. Figure 3.8 illustrates the architecture of the SSD model with the VGG backbone.

[11]Input size 300×300
[12]Frame per second
[13]Input size 512×512

Figure 3.9 shows that a small set of default boxes with different aspect ratios in two feature maps (e.g. 8×8 and 4×4 in (b) and (c)) with different scales are evaluated. For each of the default boxes in (b) and (c), the model predicts shape offsets and a confidence score. During the training, each of the default boxes is matched with the ground truth boxes like in (a) by the Jaccard overlap[14] higher than a threshold of 0.5. As it is also clear from the Figure, small objects can be captured by large fine-grained feature maps, and large objects can be captured by small coarse-grained feature maps [33].

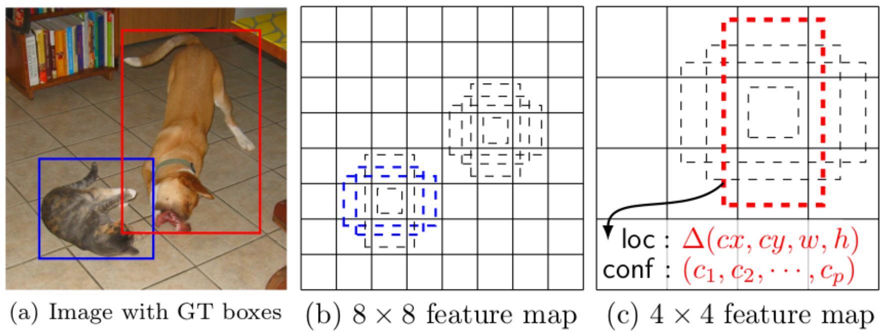

loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$
(a) Image with GT boxes
(b) 8×8 feature map
(c) 4×4 feature map

Figure 3.9: SSD framework: (a) ground truth, (b) fine-grained feature map 8×8, (c) coarse-grained feature map 4×4.

Figure from [33]

For the ALPR detection, we are using the SSD architecture embedded in Google's object detection API. It should be noted that the implementation of SSD architecture in the API is followed by the methodology section of the SSD paper, except that in the prediction part of the network, batch normalization[15] is used [37].

[14]a.k.a Intersection over Union (IoU)

[15]Batch normalization is used to reduce the effect of *internal covariate shift* and thence stabilize and accelerate the learning process.

3.4.1 The Choice of Feature Extractor

Different feature extractors have been examined as the backbone for some object detectors by Google's research team, and the result is shown in Figure 3.10. The SSD algorithm performs slightly better than the other two algorithms in terms of GPU time and satisfactorily in terms of the overall mAP. When trained with Resnet101 [40] and MobileNet [41] feature extractors, the SSD model is the most accurate of the fastest models.

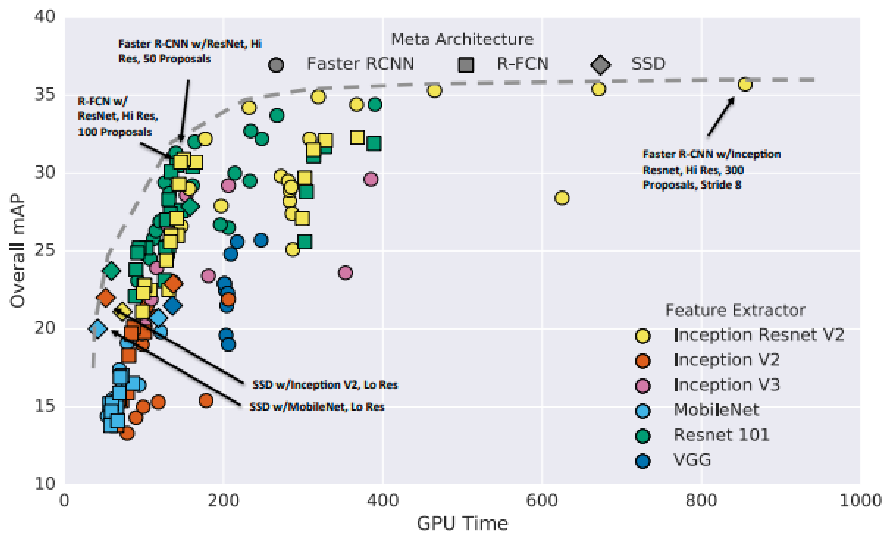


Figure 3.10: Result of different object detectors tested with different feature extractors on the COCO dataset.

Figure from [33]

Authors in [37] verified that the performance of the models in object detection is somehow correlated to the performance of the models in the classification task. Figure 3.11 indicates the accuracy of detectors on COCO versus the accuracy of the feature extractors which is measured by top-1 accuracy on ImageNet. However, the authors noticed that the SSDs performance is less sensitive to the quality of the feature extractor than the faster R-CNN and R-FCN.

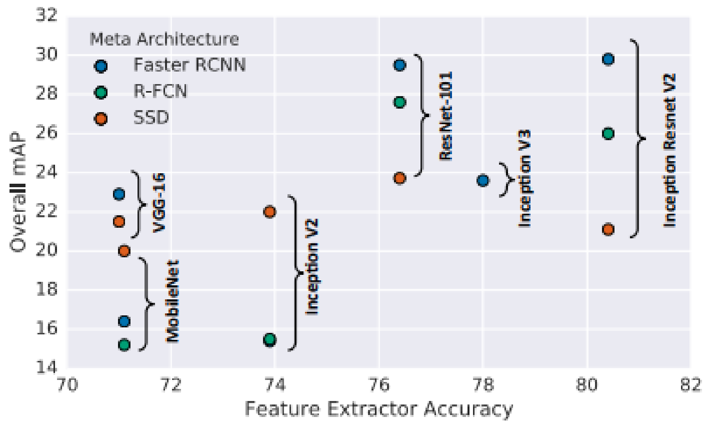


Figure 3.11: Accuracy of detector vs accuracy of feature extractor

Figure from [33]

For each object detector, one can choose a variety of feature extractor models. The choice of feature extractor is essential as the number of parameters affects the object detector's performance. Figure 3.12 shows each model's top-1 accuracy on ImageNet besides the number of parameters for each model. As the number of parameters increases, the accuracy of the model also increases. However, other factors should be taken into consideration when choosing any of the feature extractor models.

Model	Top-1 accuracy	Num. Params.
VGG-16	71.0	14,714,688
MobileNet	71.1	3,191,072
Inception V2	73.9	10,173,112
ResNet-101	76.4	42,605,504
Inception V3	78.0	21,802,784
Inception Resnet V2	80.4	54,336,736

Figure 3.12: Properties of six feature extractors used in object detection API

Figure from [33]

3.4.1.1 MobileNet-V1

The MobileNet-V1 [41] 's main feature is its implementation with the depth-wise separable convolutions block, which consists of a depth-wise convolution layer followed by a point-wise (a.k.a. 1×1) convolution layer. The depth-wise separable convolution is similar to the traditional convolution; however, it is computationally faster.

The first layer of MobileNet-V1 consists of a 3×3 standard convolution layer, followed by 13 depth-wise separable convolution layers, making it nine times faster than any comparable neural network with the same accuracy. Figure 3.13 depicts the convolution block used in MobileNet-V1 architecture.

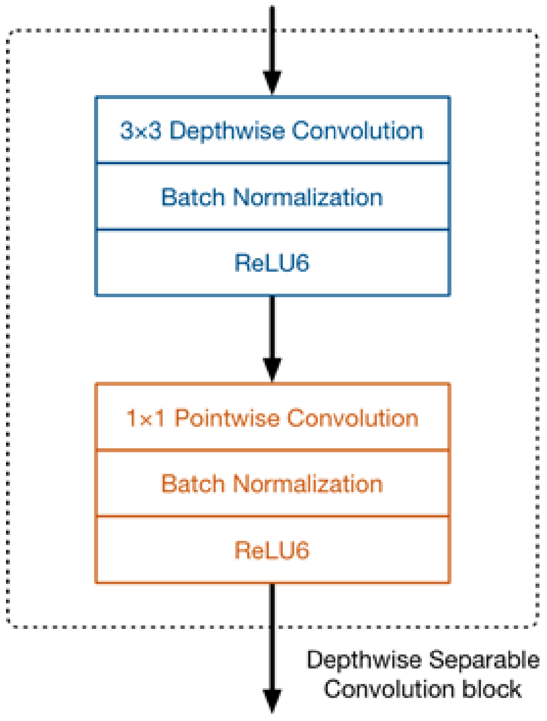

3×3 Depthwise Convolution
Batch Normalization
ReLU6
1×1 Pointwise Convolution
Batch Normalization
ReLU6
Depthwise Separable Convolution block

Figure 3.13: MobileNet-V1's Convolution block

Figure from:https://machinethink.net/blog/mobilenet-v2/

3.4.1.2 MobileNet-V2

MobileNet-V2 [42] is similar to MobileNet-V1 in terms of using depth-wise separable convolutions. However, the convolution block in MobileNet-V2 is a bit different and consists of three convolutional layers. The block in MobileNet-V2, which is shown in Figure 3.14, is called the Bottleneck Residual Block. The first layer in the block is called the expansion layer[16], and it is followed by depth-wise convolution and projection layers[17], respectively. Like ResNet architecture, MobileNet-v2 also contains the residual connection, which helps the flow of gradients through the network.

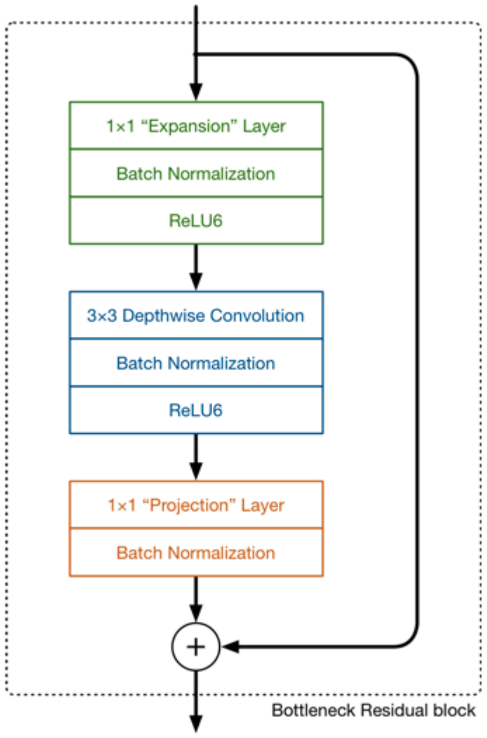


Figure 3.14: MobileNet-V2's Convolution block

Figure from:https://machinethink.net/blog/mobilenet-v2/

[16]Expands the number of channels.
[17]a.k.a. bottleneck layer and reduces the number of channels.

3.4.1.3 Inception-V2

Inception network was one of the earliest attempts of researchers in developing CNN networks. The networks before Inception models were only convolution layers stacked deeper together, making them more prone to over-fitting. The idea behind inception models is to get wider rather than deeper networks to reduce the deep neural networks' computational burden while obtaining good performance.

Authors in Inception-V2 [2] tried to make the neural network wider to reduce the loss of information, so they proposed three different modules to build the Inception-V2 network. The first module, Figure 3.15(b), is to factorize 5×5 convolution to two 3×3 convolution operations. The second module, Figure 3.15(c), is to factorize convolutions of filter size $n \times n$ to a combination of $1 \times n$ and $n \times 1$ convolutions. The third module, Figure 3.15(d), expands the filter bank to make it wider instead of deeper to lessen the loss of information.

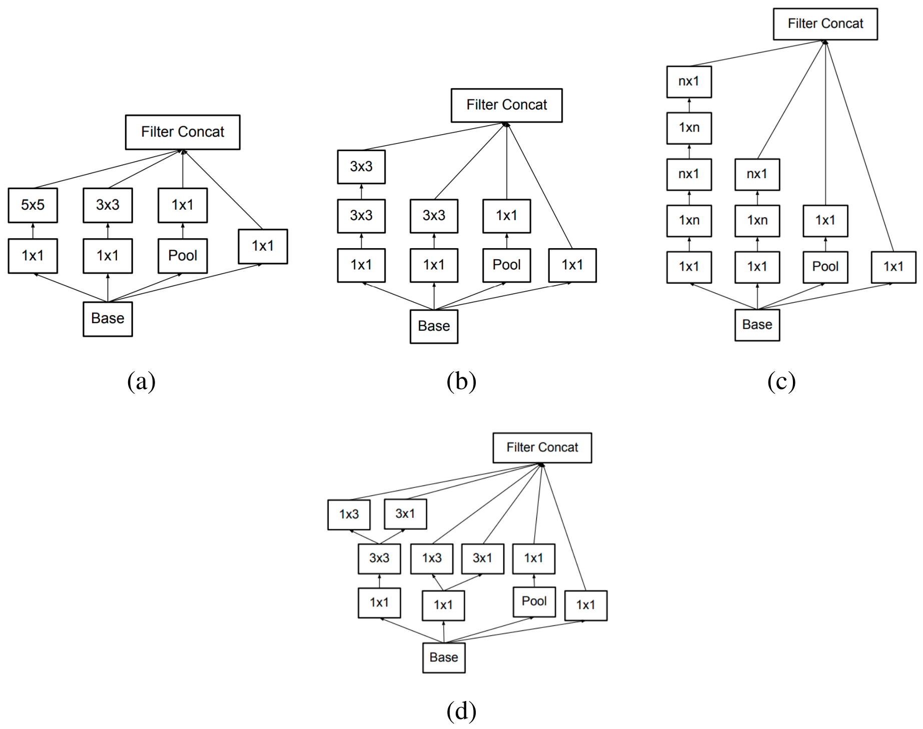


Figure 3.15: (a) Original Inception Module from [1], (b) is the first module of Inception-V2, (c) is the second module of Inception-V2, and (d) is the third module of Inception-V2. Images (b),(c), and (d) are from [2]

3.4.1.4 ResNet-50

So far, we have learned that increasing the network's depth should increase the model's accuracy. Even if the over-fitting problem is being taken care of, we will encounter the vanishing gradients problem by stacking so many layers to the neural network.

ResNet-50 is a widely used variant of ResNet [43] models. Due to the ResNet models' framework, it is possible to train deep neural networks without having any concern about vanishing gradients. The residual building block in ResNet-50/101/152 models is shown in Figure 3.16. Residual connection adds value from the beginning of the block to the end of the block and does not go through the activation functions. Hence, during the back-propagation step, the network would have higher derivatives.

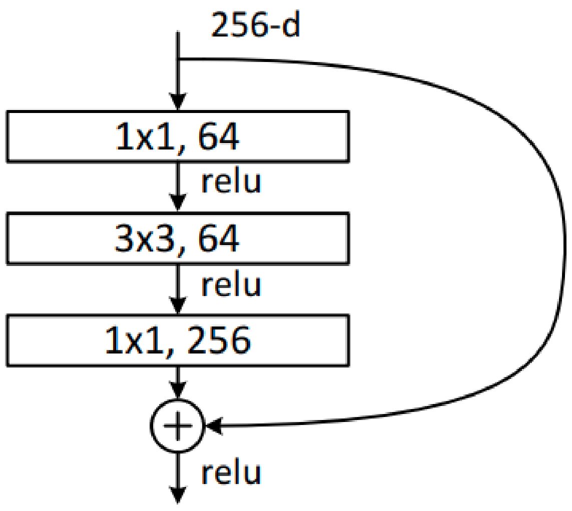


Figure 3.16: Bottleneck building block for ResNet-50

Figure from [43]

3.5 You Only Look Once (YOLOv4)

In recent years, object detectors' development was in a direction to add some layers (a.k.a. neck) between the feature extractor (a.k.a. backbone) and the detector (a.k.a. head) to collect feature maps of different stages to help increase the detection rate [44]. However, some researchers try to build a new object detection model, while others try to create a new backbone.

Another one-stage detection model is the YOLOv4 [44] which was released in April 2020. Like RCNN family, each member of the YOLO family is built upon the previous model and tried to improve the former model. The authors of YOLOv4 model use a combination of the CSPNet [45] and Darknet-53 (backbone of the YOLOv3 model) as the feature extractor for the YOLOv4 model. For the neck of the model, the authors uses the Spatial Pyramid Pooling [46] after CSPDarknet53 to increase the receptive field and separate out the most important features from the backbone. Afterward, they use Path Aggregation Network (PAN) [47] to perform feature aggregation. As for the head of the network the authors used the same YOLO head as YOLOv3 for detection. Figure 3.17 depicts the connection between backbone, neck, and head in one-stage and two-stage object detectors.

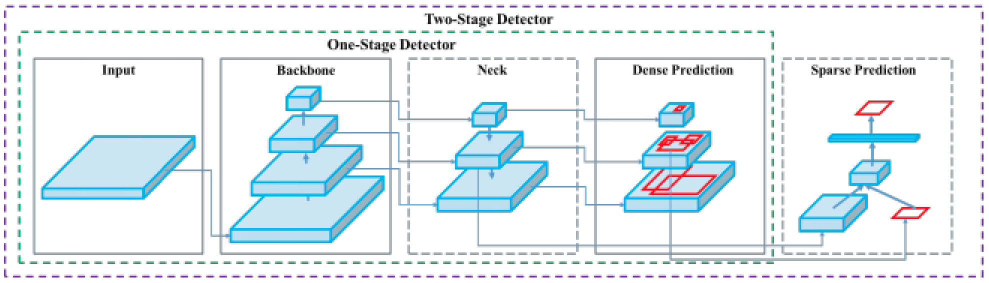


Figure 3.17: Building blocks of one-stage or two-stage object detectors.

Figure from [44]

Chapter 4

Experiments and Results

4.1 Dataset

The collection of images we use to train deep-learning models have great impact on the result. The size, resolution, and total number of images in a dataset are some of the factors that can affect the result of training a deep-learning model. The license plate datasets used during this research are UFPR-ALPR[1] and CENPARMI[2] datasets.

4.1.1 UFPR-ALPR dataset

The UFPR-ALPR [8] dataset consists of 4500 images from 150 vehicles (150 videos with a duration of 1 second and frame rate of 30 FPS), captured by GoPro Hero4 Silver, Huawei P9 Lite, and iPhone 7 Plus cameras. The dataset, table 4.1, is split as 40 % training,20 % for validation, and 40 % for testing. The split is done in a way that every split has the same distribution of images. Figure 4.1 shows some examples of this dataset.

[1]https://web.inf.ufpr.br/vri/databases/ufpr-alpr/

[2]Center for Pattern Recognition and Machine Intelligence

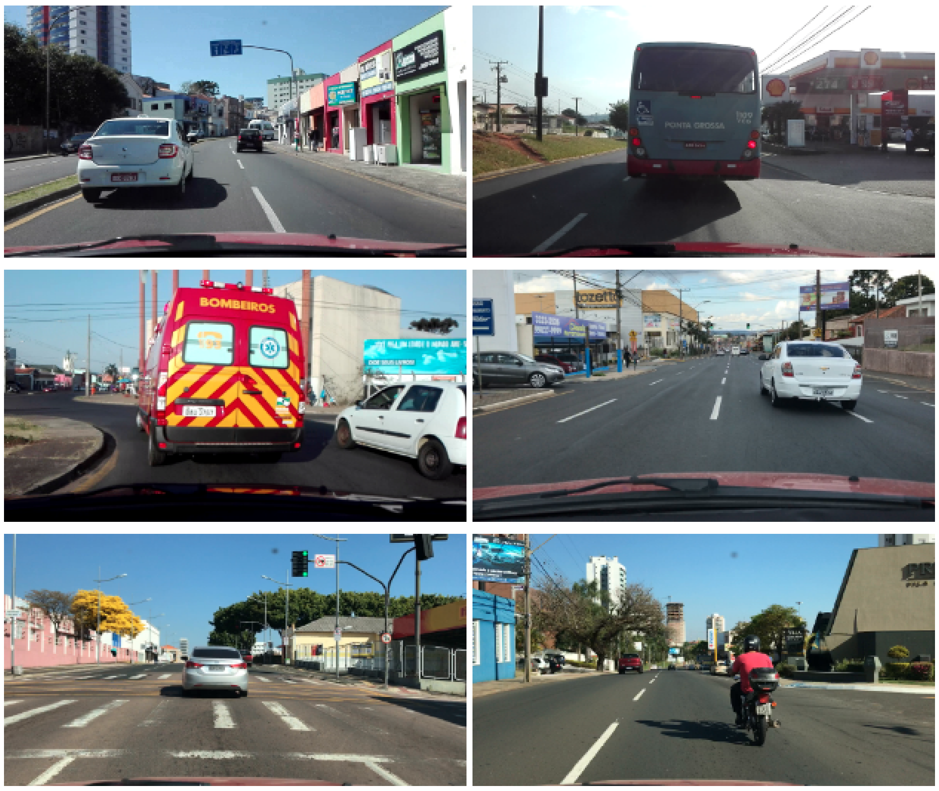


Figure 4.1: Example of UFPR-ALPR dataset

4.1.2 CENPARMI dataset

The CENPARMI dataset consists of license plate images from the United States and Canada, captured by iPhone 6, Nexus 5X, Canon, and Sony cameras. The images are annotated with the LabelImg [48] tool by the author of this research. First, 10 % of the whole dataset, table 4.1, is test data, and from the remaining images, the train and validation split created (80 % of the remaining for train data and 20 % of remaining for validation). Figure 4.2 represents some examples from the CENPARMI dataset.

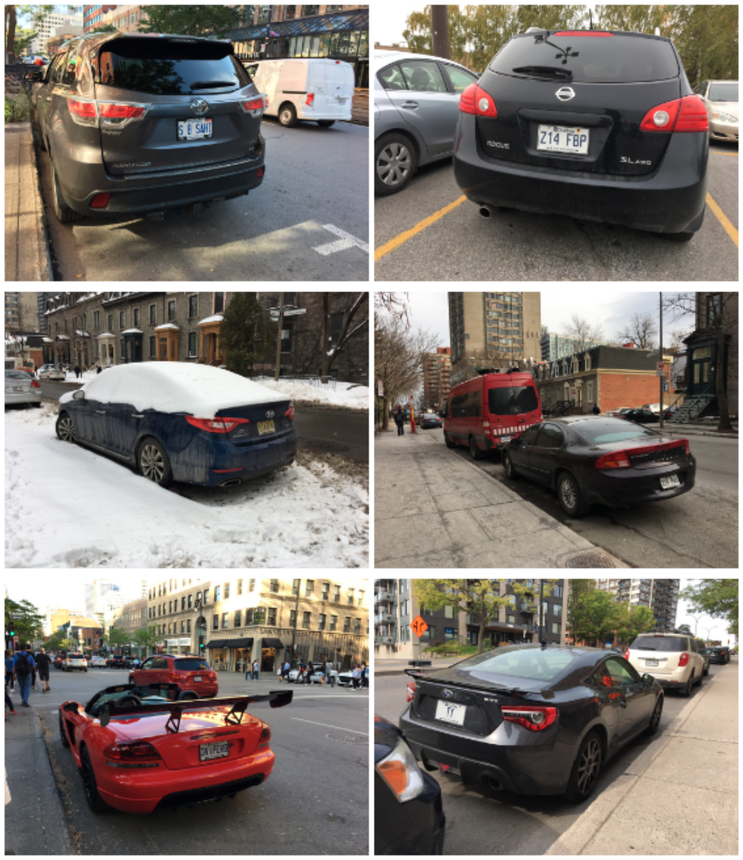


Figure 4.2: Example of CENPARMI dataset

Dataset	Train	Validation	test	total
UFPR-ALPR	1800	900	1800	4500
CENPARMI	1486	372	206	2064

Table 4.1: The datasets and the train, validation, and test splits.

4.1.3 Pre-processing

As mentioned earlier, the amount of data and resolution are among the factors that affect the performance of a deep learning model. For the UFPR-ALPR dataset to be comparable with other works, we decided to work with it as is. However, the CENPARMI dataset needed to be refined in order to be fed to a neural network. The CENPARMI dataset consisted of 1270 images of license plates from the US and Canada with very high resolution. After the cleaning and downsampling process, the number of images in this dataset changed to 1006. Figure 4.3 shows some images of license plate that have been removed from the dataset.

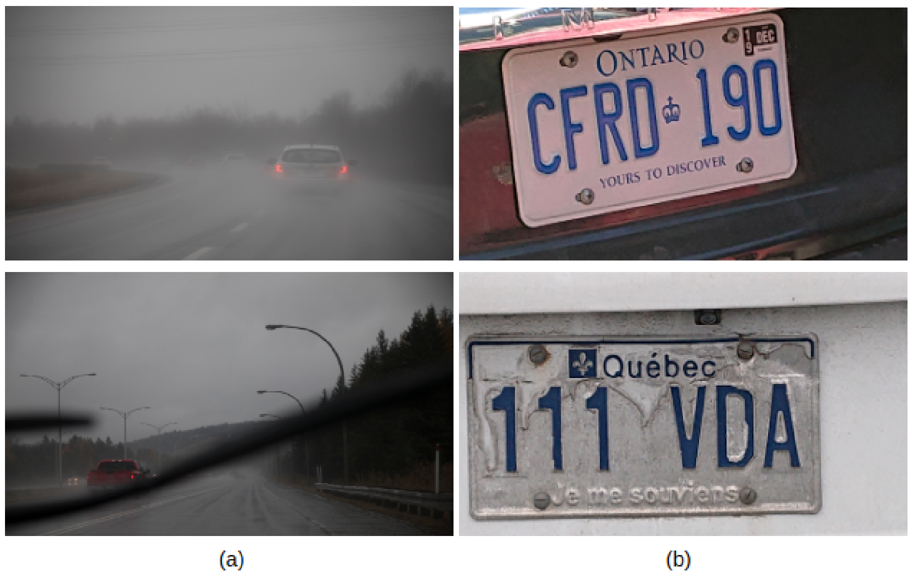
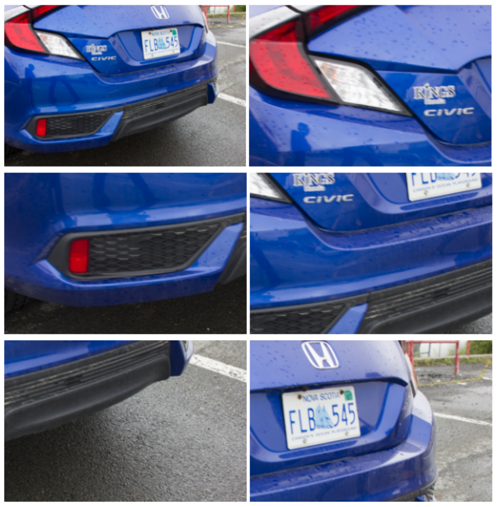


Figure 4.3: Example of removed images from CENPARMI dataset. License plates in (a) are not visible, and license plates in (b) are too close to the camera.

The number of images in CENPARMI dataset is not sufficient when we deal with deep-learning models which need a lot of data for promising results. Hence, to increase the number of images, we cropped every image relative to its width and height into five regions of top-right, bottom-right, top-left, bottom-left, and centre. Figure 4.4 illustrates two samples with their five crops.


(a)

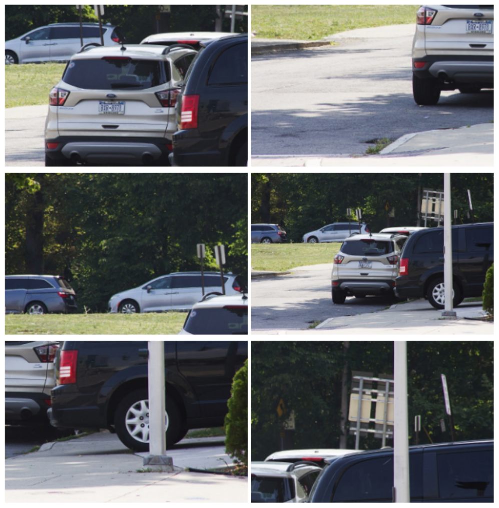
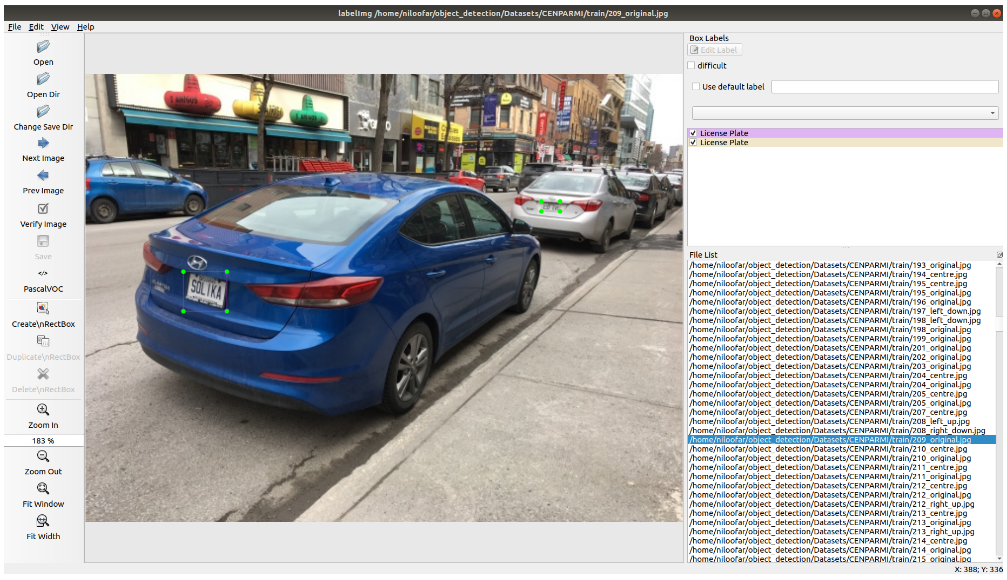

(b)

Figure 4.4: (a) and (b) are two examples of cropped images from CENPARMI dataset into 5 regions.

During the image cropping process, some images without any license plates were generated. Therefore, we were required to clean the dataset again. After all, the total number of images in the CENPARMI dataset added up to 2064. Moreover, we labeled all the images with the LabelImg tool as is shown in Figure 4.5.

Figure 4.6 shows the ground-truth bounding box width versus height of the training data. The slope shows how different are the aspect ratios of the bounding boxes. From the Figure, we can acquire that the License plates in UFPR-ALPR dataset are a tiny part of the images, and two separate mass of orange dots show car's license plates, which are rectangular, vs. motorcycle's license plates, which are squarish. In contrast, the license plates in the the CENPARMI dataset have various sizes, and this is due to the way the images are taken from the vehicles and not that the license plates themselves are of various sizes in the North American countries.



labelImg /home/niloofar/object_detection/Datasets/CENPARMI/train/209_original.jpg

✓ License Plate
✓ License Plate

```
/home/niloofar/object_detection/Datasets/CENPARMI/train/193_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/194_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/195_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/195_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/196_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/197_left_down.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/198_left_down.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/198_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/199_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/201_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/202_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/203_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/204_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/204_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/205_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/205_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/207_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/208_left_up.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/208_right_down.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/209_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/210_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/210_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/211_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/211_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/212_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/212_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/212_right_up.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/213_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/213_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/213_right_up.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/214_centre.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/214_original.jpg
/home/niloofar/object_detection/Datasets/CENPARMI/train/215_original.jpg
```

X: 388; Y: 336

Figure 4.5: Example of annotating an image with LabelImg tool.

4.2 Experiment

We experimented two different object detection models with CENPARMI and UFPR-ALPR datasets. SSD model with MobileNetV1, MobileNetV2, InceptionV2, and ResNet50 feature extractors and the very recent YOLOv4 model have investigated in this project. The reason for choosing the above mentioned feature extractors rather than the others is that they have less number of parameters, thus they are faster and more acceptable for the purpose of our project.

Moreover, having a limited amount of data leads us to choose feature extractors with fewer parameters and networks[3] pre-trained on the COCO dataset to avoid over-fitting for both SSD and YOLOv4 object detection models. There are several methods to reduce the chance of over-fitting, such as data augmentation, feature selection, L1/L2 regularization, early stopping, and dropout, each of which approaches the over-fitting problem differently.

[3]Network is implemented using the TensorFlow Detection Model Zoo framework: `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md`

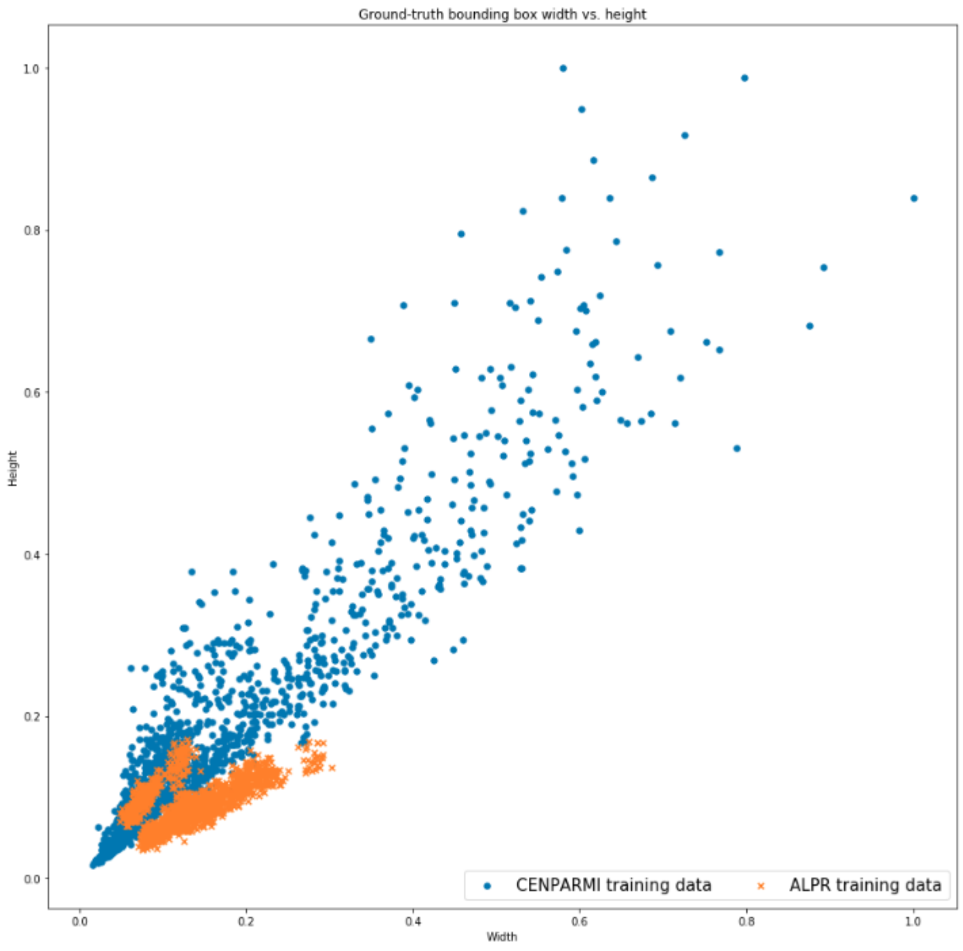


Figure 4.6: Ground-truth bounding box width versus height of training data for CENPARMI and UFPR-ALPR datasets

In our experiment on SSD models, we have used data augmentation, batch normalization, and L2 regularization methods to handle the over-fitting issue. As data augmentation, we used *random horizontal flip*, *random crop*, and *random patch Gaussian* to make our datasets larger and help the networks to generalize better. Batch Normalization is used to standardize the input to layers and reduce the number of training epochs. As for regularization method, We used L2 regularization with weight decay of 4×10^{-5} to reduce the possibility of over-fitting.

We scrutinized the SSD model more because we believed that the SSD model performs better in detecting smaller objects than YOLO models (up to YOLOv3) when performing this project. The SSD model takes feature maps of different stages to perform detection; the number of parameters it consists of when used with very light feature extractors leads to better results. However, after the release of YOLOv4 in April 2020, we decided to train our datasets and check the new model's result. The improvement done on YOLOv4 is enormous, and the result we got on datasets are promising.

The license plate detection with SSD model is performed through the TensorFlow Object Detection API[4] which is built on top of TensorFlow framework for localizing objects. On the other hand, YOLOv4[5] is performed through the OpenCV framework. The SSD models evaluated on-premise on octa-core Intel Core i7-9700 with 16 GB RAM, and GeForce RTX 2070 GPU while the YOLOv4 model evaluated on Google Colab[6] which is a cloud service.

A lot of experiments and hyper-parameter tuning have been conducted throughout this research to examine the SSD and YOLOv4 models and get favorable results on the CENPARMI and UFPR-ALPR datasets. The Table 4.2 illustrates the main configurations we used to achieve the final results, and Figure 4.7 demonstrates the steps we took from installing libraries and dependencies to making predictions. In the following section, we will discuss the results.

	CENPARMI / UFPR-ALPR			
Models	Input Size	Optimizer	Batch size	Learning rate
MobileNetV1+SSD	300×300 / 640×640	Adam / Momentum	64/8	0.008/0.0004
MobileNetV2+SSD	300×300 / 600×600	RMSprop/Adam	24/10	0.004/0.01
InceptionV2+SSD	300×300 / 600×600	RMSprop	24/12	0.004/0.0004
ResNet50+SSD	640×640 / 640×640	Momentum	4/2	0.001/0.004
YOLOv4	416×416 / 416×416	Momentum	64/64	0.001/0.001

Table 4.2: The input size, optimizer, batch size, and learning rate of models trained on CENPARMI and UFPR-ALPR datasets.

[4]https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/
[5]https://github.com/AlexeyAB/darknet
[6]https://colab.research.google.com/drive/12QusaaRj_lUwCGDvQNfICpa7kA7_a2dE

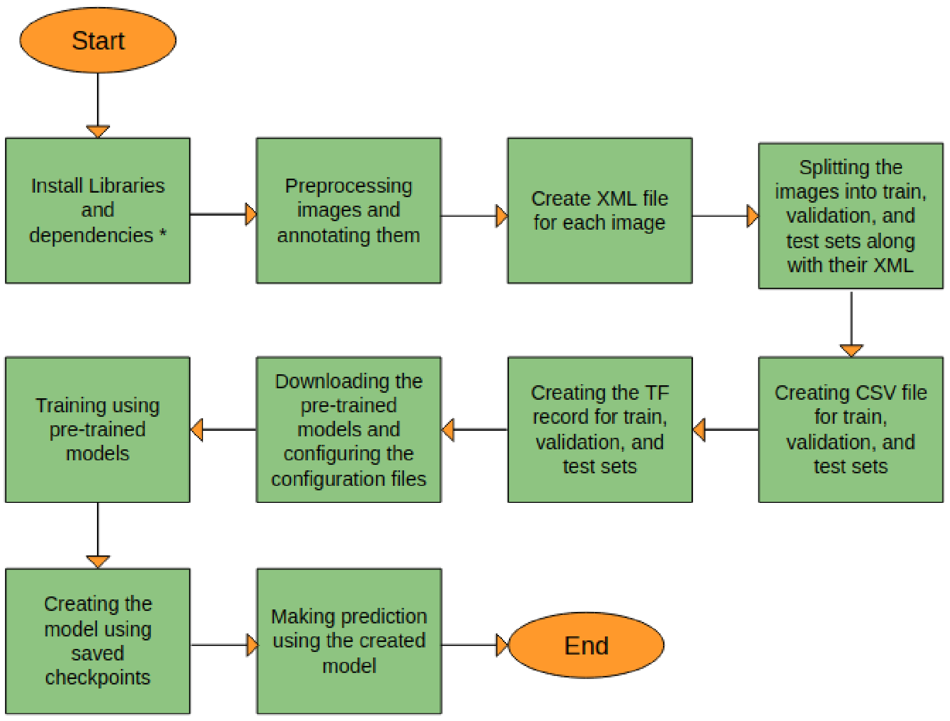

Start
Install Libraries and dependencies *
Preprocessing images and annotating them
Create XML file for each image
Splitting the images into train, validation, and test sets along with their XML
Creating CSV file for train, validation, and test sets
Creating the TF record for train, validation, and test sets
Downloading the pre-trained models and configuring the configuration files
Training using pre-trained models
Creating the model using saved checkpoints
Making prediction using the created model
End

Figure 4.7: Object Detection Work Flow.

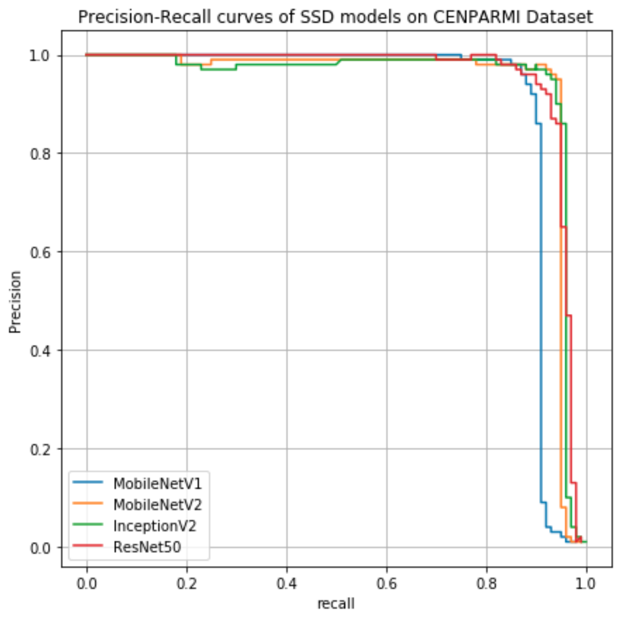
*https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/

4.3 Results

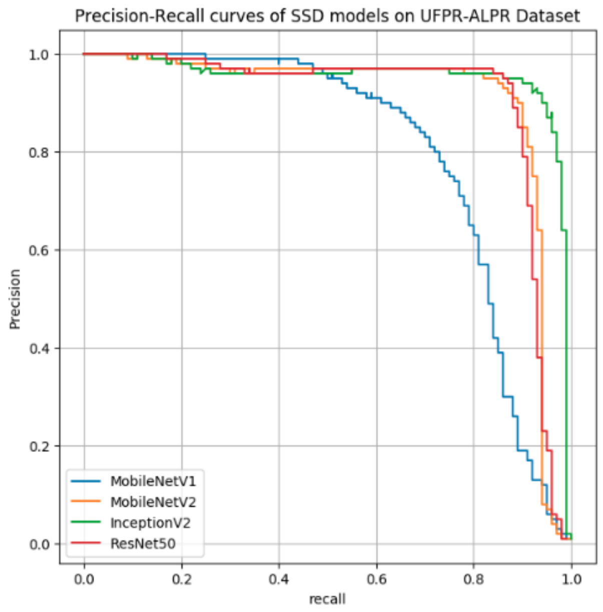
Precision-recall curves are useful when we are trying to decide which model is appropriate for our needs. Some applications need a system with few predictions but mostly correct (High Precision - Low Recall). Some other applications require numerous results, while few are correct (High Recall - Low Precision). In the latter, results are more likely to contain false positives.

An ideal system is the one with high precision and high recall, in which there are plenty of predicted labels with all results labeled correctly. As the precision-recall curve in Figure 4.8(a) illustrates, all of the four SSD models perform reasonably well on the CENPARMI dataset.

However, Figure 4.8(b) shows that the mentioned models perform slightly differently on the UFPR-ALPR dataset. The difference between the two datasets and the size of license plates in the UFPR-ALPR dataset concerning each image's total size makes the feature extraction and hence detection a bit harder, which means we do need to do more hyperparameter tuning if we want to have better results than what we already have.



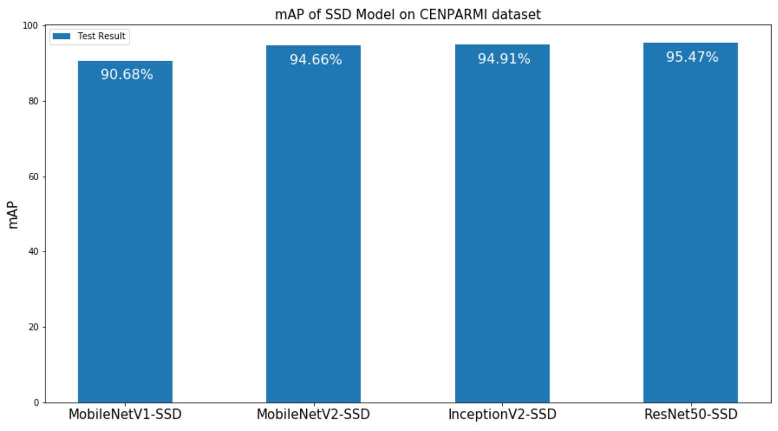
(a)



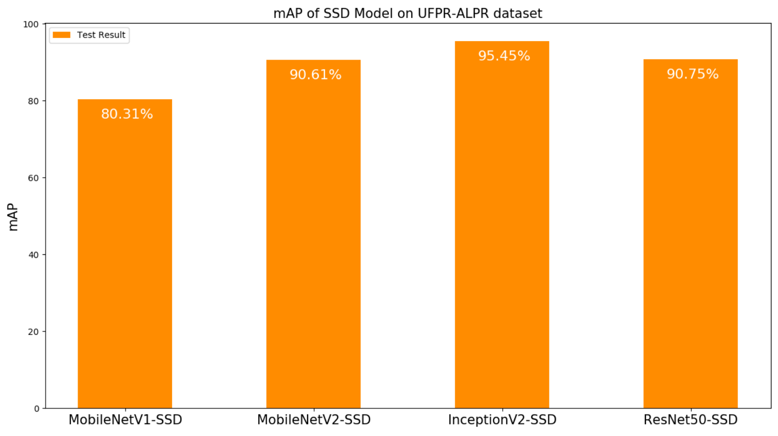
(b)

Figure 4.8: (a) Precision-recall curve of models on the CENPARMI dataset, (b) Precision-recall curve of models on the UFPR-ALPR dataset.

Although the precision-recall curve can show us the detectors' performance, having a numerical value is also of great importance. Figure 4.9 shows the mAP of the test data on the trained SSD models. Figure 4.9(a) is the test result of the CENPARMI dataset (the blue bar chart), and Figure 4.9(b) is the test result of the UFPR-ALPR dataset (the orange bar chart).



(a)



(b)

Figure 4.9: (a) Test result of the CENPARMI dataset, (b) Test result of UFPR-ALPR dataset.

We examined the SSD model with four different feature extractors, and as it is clear from Figure 4.9, the result of mAP is more promising when the model has trained with feature extractors with a higher number of parameters. Table 4.3 lists all the feature extractors with their classification accuracy and the number of parameters. From Figure 4.9 and table 4.3, we can infer a correlation between the number of parameters, classification accuracy, speed of a specific feature extractor, and the detection performance. The higher the number of parameters of a feature extractor, the higher the classification accuracy and lower the speed.

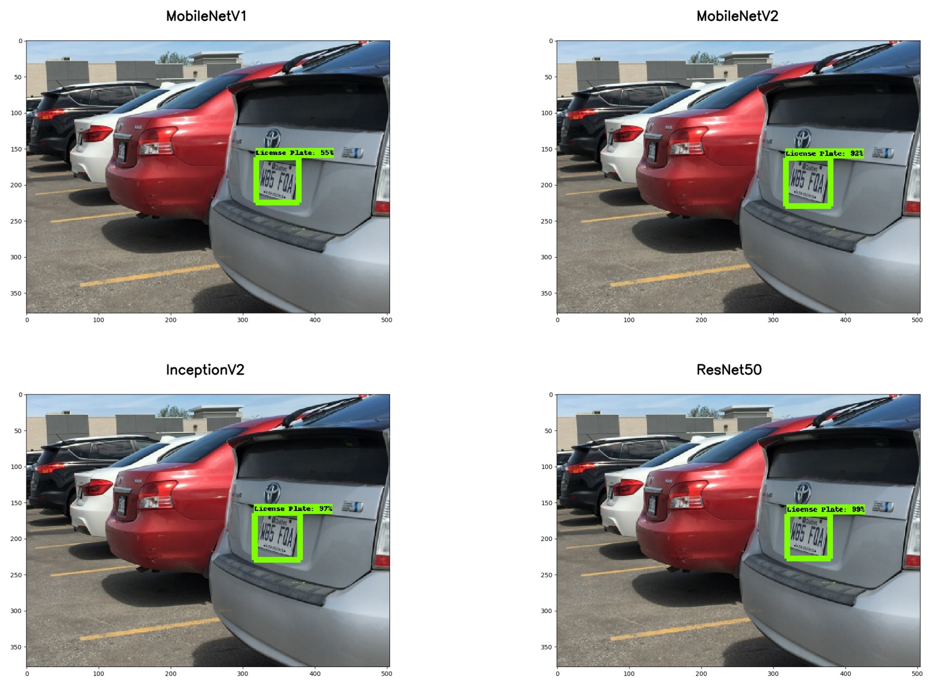
Feature Extractor	**Classification Accuracy top-1/top-5**[*]	**mParam**	**Speed (ms)**
MobileNetV1	71.03 %/89.94 %	4.221	30 (GTX TITAN X)
MobileNetV2	74.09 %/91.97 %	6.087	31 (GTX TITAN X)
InceptionV2	74.084 %/91.798 %	11.185	58 (GTX TITAN X)
ResNet50	76.17 %/92.98 %	25.53	89 (GTX TITAN X)
CSPDarknet53	-	27.6	65 (Tesla V100)

[*] Top-N accuracy means that the correct class gets to be in the Top-N probabilities for it to count as “correct”.

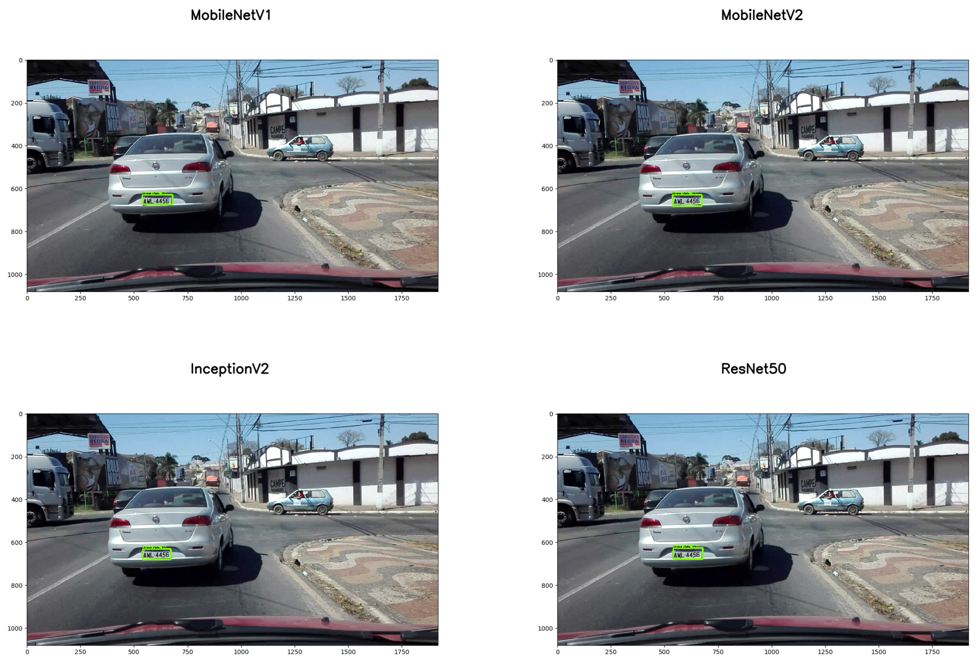
Table 4.3: Classification accuracy, number of parameters, and speed of each feature extractor reported by OpenVINO and Object Detection API.

In addition, as it is clear from the Figure 4.9(b), we expected that the SSD-ResNet50 model gives a better result than SSD-InceptionV2; however, the fact is that we should do more hyperparameter tuning for the SSD-ResNet50 on the UFPR-ALPR dataset[7]. Figure 4.10 shows the test result of the above mentioned feature extractors used in SSD model on test images of CENPARMI and UFPR-ALPR datasets.

[7]The combination of images with high resolution for training in the UFPR-ALPR dataset and the number of SSD-ResNet50 model’s parameters led us to face memory issues.



(a)



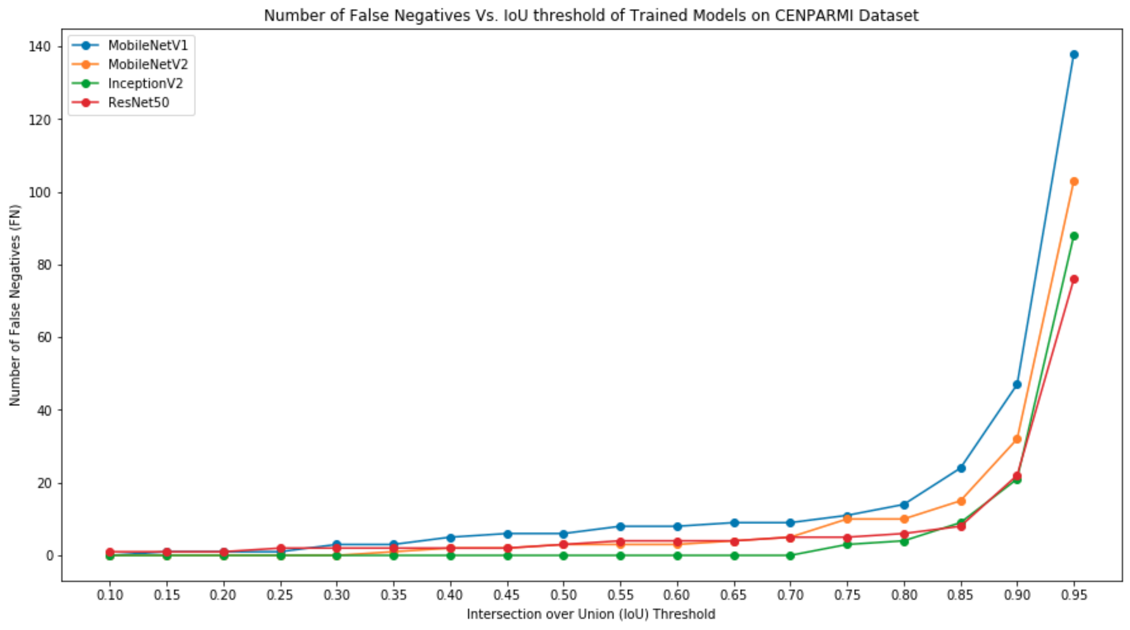
(b)

Figure 4.10: (a) is the test result of SSD model with different feature extractors on CENPARMI dataset, (b) is the test result of SSD model with different feature extractors on UFPR-ALPR dataset.

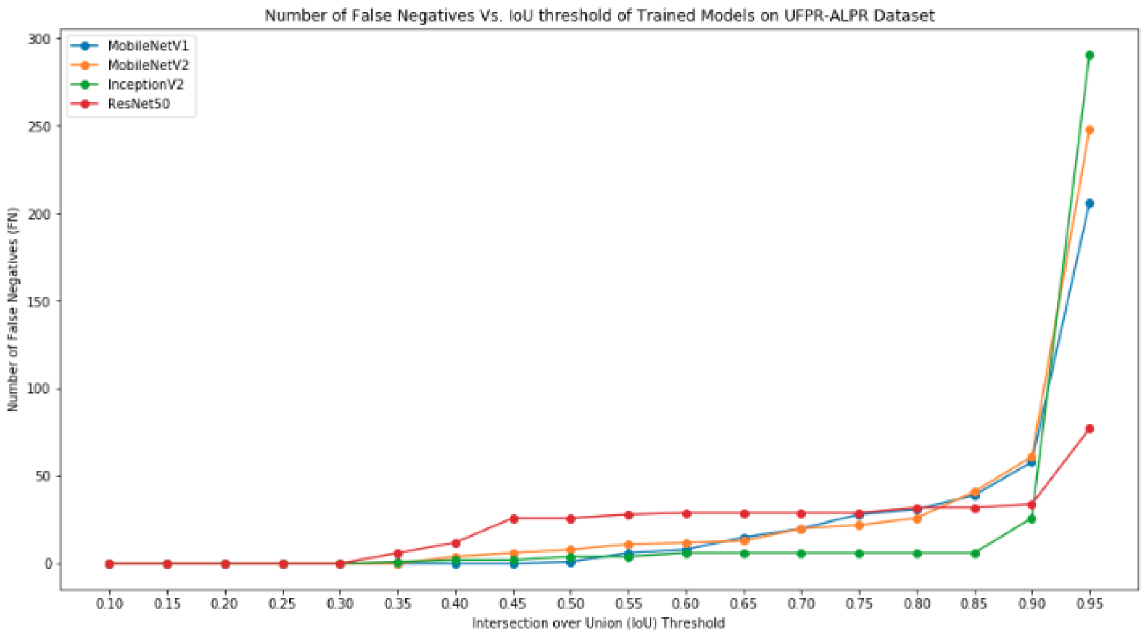
In the test set of the CENPARMI dataset, there are images with more than one license plate in them; however, mostly the SSD-InceptionV2 and SSD-ResNet50 could not detect at least one of the license plates like in Figures 4.12. Again this does not mean SSD-InceptionV2 and SSD-ResNet50 are performing poorly, but that they are deeper networks and need more tuning.

Deeper networks should work better than shallower networks; however during the course of this experiment, we have seen that deep feature extractors did not perform as expected in detecting license plates. This does not mean shallow networks perform better than deep networks in general, only that deep networks are harder to optimize [1].

Figure 4.11 shows the number of False Negatives (FN) concerning the Intersection over Union (IoU) for CENPARMI and UFPR-ALPR datasets. The higher number of False Negatives corresponds to the lower recall, and as the IoU increases, recall decreases, which shows the effectiveness of detection proposals.

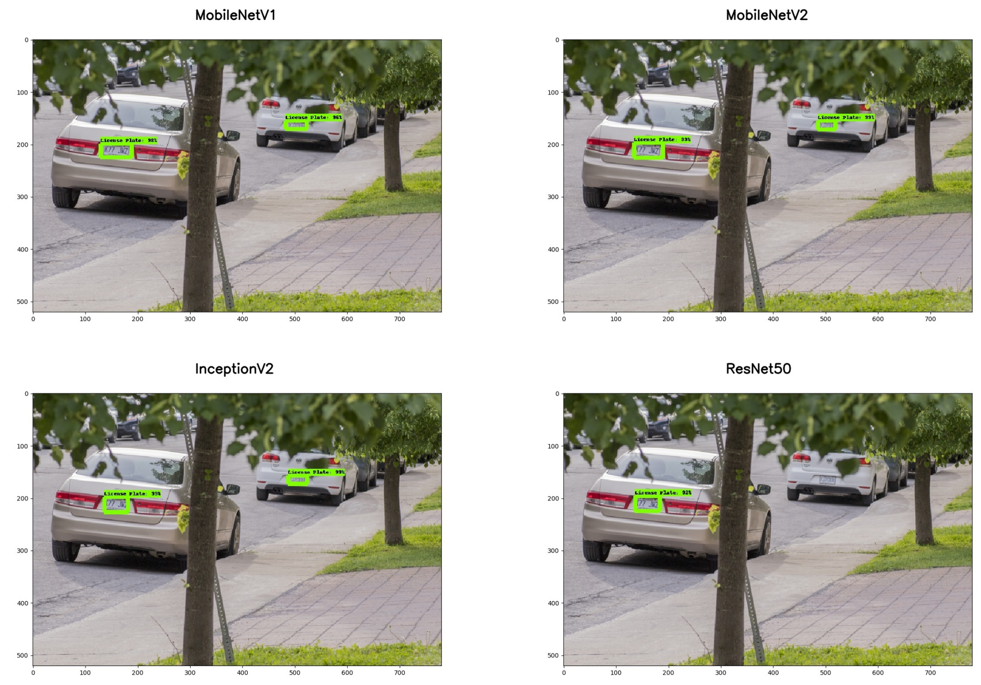


(a)

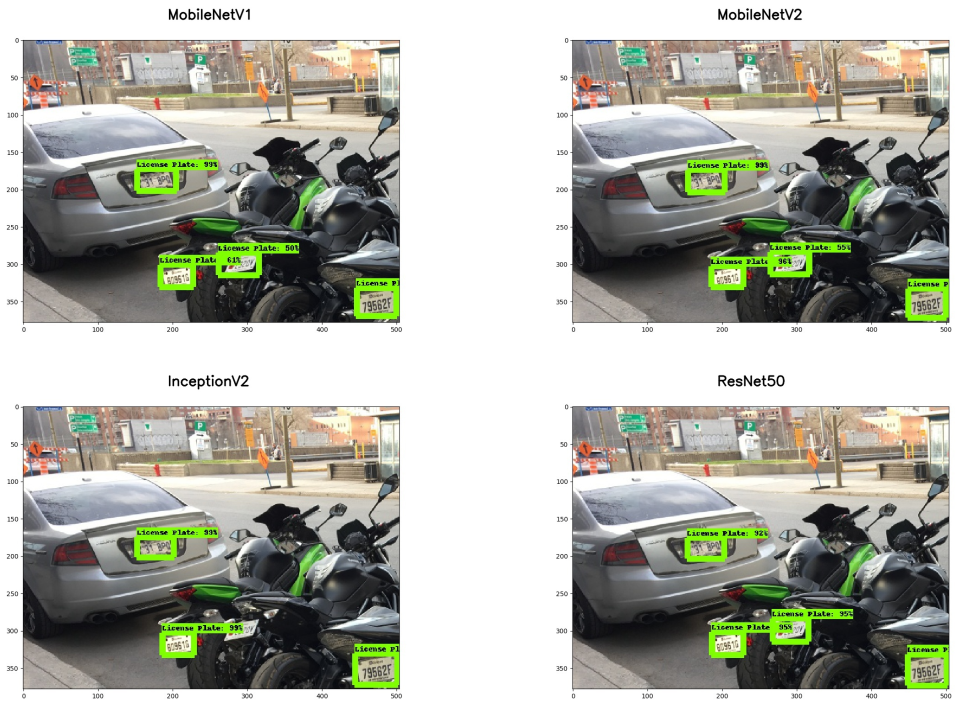


(b)

Figure 4.11: (a) False Negatives vs IoU on the CENPARMI dataset, (b) False Negatives vs IoU on the UFPR-ALPR dataset.



(a)



(b)

Figure 4.12: Test data from CENPARMI dataset showing that (a) SSD-ResNet50 could not detect the second license plate in the image, (b) SSD-InceptionV2 could not detect the license plate in the image.

The CENPARMI and UFPR-ALPR datasets have been previously tested with the YOLOv2 detection model by the CENPARMI lab's researchers, which also presented an acceptable result on the mentioned datasets. However, in this study we have decided to evaluate a wider range of fast detection models to better conclude the need for an ALPR system. Consequently, we have tested two of the fast detection models, the SSD model with different feature extractors as explained earlier, and YOLOv4. The result of each model is shown in Figure 4.13.

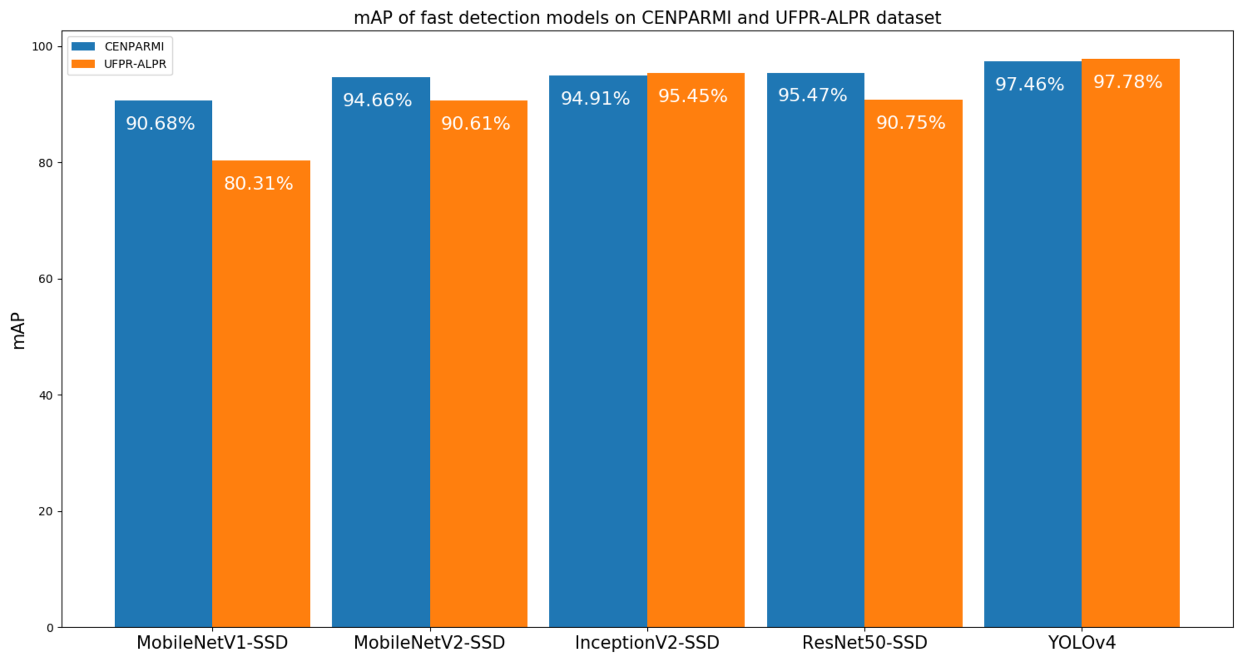


Figure 4.13: The result of CENPARMI and UFPR-ALPR test sets on different fast detection models.

As it is clear from the Figure 4.13, the SSD model performed better on the CENPARMI dataset than in UFPR-ALPR dataset. The license plates in UFPR-ALPR dataset encompass a very small portion of images and this is one of the reasons that the performance of SSD model is not as good as the performance on CENPARMI dataset.

As mentioned earlier the SSD model uses the upper layers for detection of fine objects and lower layers for detection of coarse objects. This feature helps to detect objects of different scales and is one of the earliest attempts of pyramidal feature hierarchy. However, SSD model builds the pyramid from higher layers of the network. Hence, it avoids low level features and then loose the chance of using high resolution feature maps [49]. Thus, the SSD model performs insufficiently when there are very small objects in the image.

Moreover, as it is clear from the Figure 4.13, the YOLOv4 model performance is slightly better in both datasets and one reason is that the authors in YOLOv4 used Path Aggregation Network (PANet) [47] which is based on the Feature Pyramidal Network (FPN)[49]. PANet is a method for boosting information flow from all feature levels which can help detection models perform better.

During this experiment, we had difficulty training some of the models, especially on the UFPR-ALPR datasets. License plates in the UFPR-ALPR dataset are a tiny proportion of the images; hence training models on this dataset were trickier. Figure 4.14 shows different attempts that we took to train the MobileNetV2-SSD on the UFPR-ALPR dataset. To get better results, we performed some hyper-parameter tuning. The main hyper-parameters that we tuned were learning rate, optimizer, and batch size.

For example, for attempt#1 to attempt#6, we used the RMSprop optimizer, but we kept changing the model's learning rate and batch size, and for attempt#7 and attempt#8, we used the Adam optimizer and tried these two models with different learning rates. Finally, as shown in Figure 4.14, our attempt on MobileNetV2-SSD with the Adam optimizer resulted in fewer false negatives, which means the model successfully detected most of the ground-truth labels.

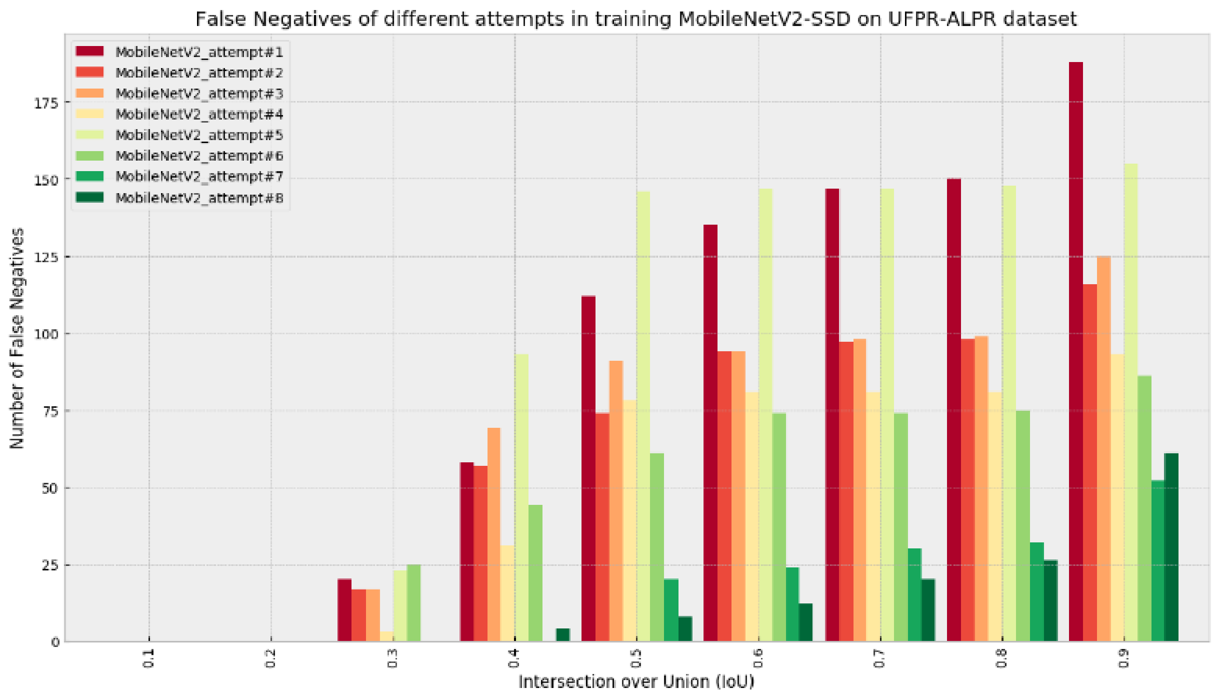


Figure 4.14: Different attempts to train the MobileNetV2-SSD model on the UFPR-ALPR dataset and the number of false negatives corresponding to each model.

Figure 4.15 shows the number of false negatives of the attempts mentioned above at the IoU threshold 0.5 and their corresponding mean average precision (mAP) results. As shown in the figure, the model in attempt#4 has the highest mAP; however, its number of false negatives is high. Depending on the application, one can choose this model or the other. Nevertheless, It is always preferable to have fewer false negatives in the ALPR system alongside high precision. In attempt#8, the model with 90.61 % mAP and eight false negatives is desirable than the model in attempt#4 with a precision of 91.36 % and 78 false negatives.

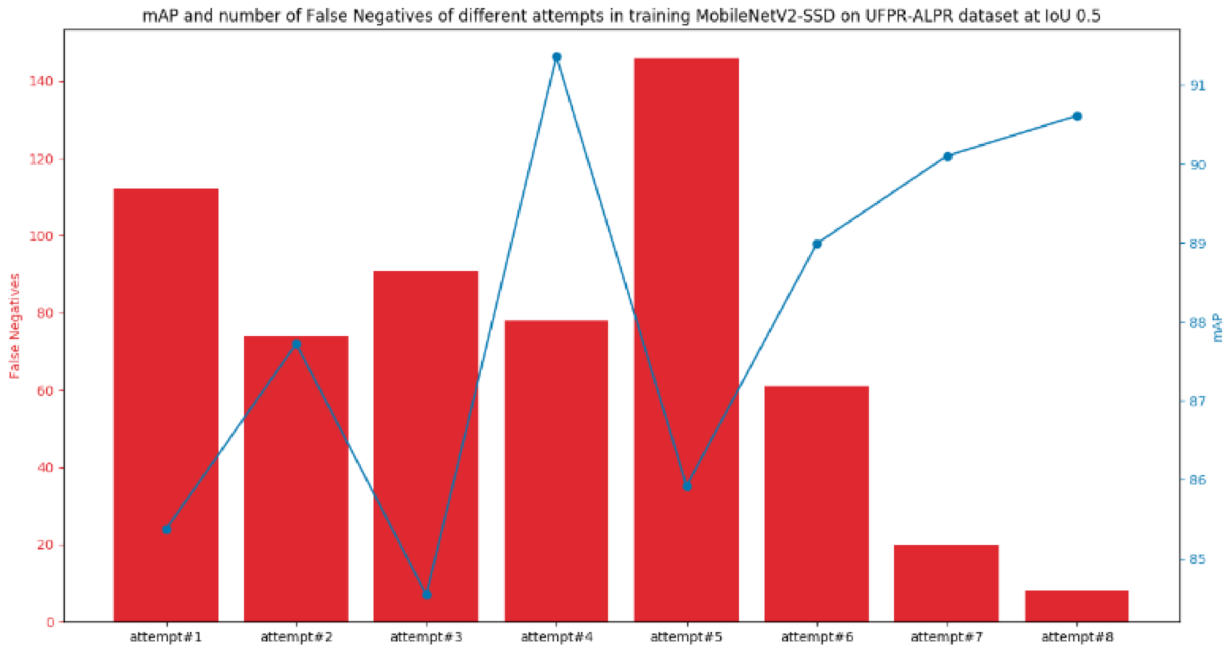


Figure 4.15: Number of false negatives of MobileNetV2-SSD models trained on UFPR-ALPR dataset at IoU threshold 0.5 and their corresponding mAP results.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Automatic License Plates Detection and Recognition (ALPR) systems are useful for different purposes, such as helping law enforcement, traffic control, and parking lot security. ALPR systems mostly deal with complex scenes, whether in real-time or offline mode. Dust, rain, illumination, and many other environmental factors, and the fact that license plates are usually a small proportion of input video or image, can make the situation a bit harder for the ALPR systems.

Object detection is the first and foremost step in ALPR, which we investigated in this project. In object detection, high precision is not the only essential criterion anymore. Nevertheless, leveraging a model that can balance accuracy, speed, and memory is crucial.

Since the ALPR systems are mostly useful in real-time applications, we evaluated the SSD and YOLOv4 models known as fast detection models on CENPARMI and UFPR-ALPR datasets. Feature extraction is a sub-task of object detection; hence we examined different feature extractors with the SSD model and found out that algorithms perform differently over different datasets [50]. We received good results with both algorithms on both datasets, although the SSD model results with different feature extractors varied a lot on the UFPR-ALPR dataset.

We learned that the higher the number of parameters of a model, the better the detection results. On the other hand, the number of parameters of a model can affect an object detection task's speed. Although we trained the selected models on pre-trained networks, we presume that more data can help the model generalize better and provide better results.

Accordingly, to have the desired ALPR system, one should consider the dataset characteristics, the model, and the hardware system for accuracy, speed, and memory performance.

5.2 Limitations and Future Work

During this experiment, we dealt with some limitations with inadequate on-premise hardware systems and datasets. In training Deep Learning algorithms, it is always better to have a bigger amount of data. The more the amount of data, the better the model can learn.

With the advent of technology and hence the emergence of autonomous vehicles, having a reliable real-time ALPR system is crucial. To further extend this work, we propose to evaluate and compare other fast detection algorithms with various other populated datasets.

We also recommend that it is beneficial to evaluate each model's memory usage and speed during the further investigation of fast detection algorithms. Moreover, for the next step, we propose to add a model such as a Recurrent Neural Network (RNN) for the recognition phase on top of the fast detection models and to evaluate each model's performance on various datasets.

Bibliography

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[3] M. Biswas and H. Gore. (2012) Best practices guide for improving automated license plate reader effectiveness through uniform license plate design and manufacture. [Online]. Available: https://www.aamva.org/WorkArea/DownloadAsset.aspx?id=2911

[4] P. Wu and Y. Lin, "Research on license plate detection algorithm based on ssd," in *Proceedings of the 2nd International Conference on Advances in Image Processing*, 2018, pp. 19–23.

[5] M. Peker, "Comparison of tensorflow object detection networks for licence plate localization," in *2019 1st Global Power, Energy and Communication Conference (GPECOM)*. IEEE, 2019, pp. 101–105.

[6] X. Peng, L. Wen, D. Bai, and B. Peng, "Reformative vehicle license plate recognition algorithm based on deep learning," in *International Conference on Cognitive Systems and Signal Processing*. Springer, 2018, pp. 243–255.

[7] L. Xie, T. Ahmad, L. Jin, Y. Liu, and S. Zhang, "A new cnn-based method for multi-directional car license plate detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 507–517, 2018.

[8] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the yolo detector," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–10.

[9] S. M. Silva and C. R. Jung, "Real-time brazilian license plate detection and recognition using deep convolutional neural networks," in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2017, pp. 55–62.

[10] B. Dhedhi, P. Datar, A. Chiplunkar, K. Jain, A. Rangarajan, and J. Kundargi, *Automatic License Plate Recognition Using Deep Learning: Third International Conference on Intelligent Information Technologies, ICIIT 2018, Chennai, India, December 11–14, 2018, Proceedings*. Springer, 01 2019, pp. 46–58.

[11] Y. Kessentini, M. D. Besbes, S. Ammar, and A. Chabbouh, "A two-stage deep neural network for multi-norm license plate detection and recognition," *Expert Systems with Applications*, vol. 136, pp. 159–170, 2019.

[12] T. K. Cheang, Y. S. Chong, and Y. H. Tay, "Segmentation-free vehicle license plate recognition using convnet-rnn," *arXiv preprint arXiv:1701.06439*, 2017.

[13] H. Li and C. Shen, "Reading car license plates using deep convolutional neural networks and lstms," *arXiv preprint arXiv:1601.05610*, 2016.

[14] H. Li, P. Wang, and C. Shen, "Toward end-to-end car license plate detection and recognition with deep neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1126–1136, 2018.

[15] X. Wang, Z. Man, M. You, and C. Shen, "Adversarial generation of training examples: applications to moving vehicle license plate recognition," *arXiv preprint arXiv:1707.03124*, 2017.

[16] S. Z. Masood, G. Shu, A. Dehghan, and E. G. Ortiz, "License plate detection and recognition using deeply learned convolutional neural networks," *arXiv preprint arXiv:1703.07330*, 2017.

[17] V. Jain, Z. Sasindran, A. Rajagopal, S. Biswas, H. S. Bharadwaj, and K. Ramakrishnan, "Deep automatic license plate recognition system," in *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*. ACM, 2016, p. 6.

[18] Z. Selmi, M. B. Halima, and A. M. Alimi, "Deep learning system for automatic license plate detection and recognition," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 1132–1138.

[19] Z. Xu, W. Yang, A. Meng, N. Lu, H. Huang, C. Ying, and L. Huang, "Towards end-to-end license plate detection and recognition: A large dataset and baseline," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 255–271.

[20] W. Nicholson. (2011) License plate fonts of the western world: History, samples, and download info. [Online]. Available: https://www.leewardpro.com/articles/licplatefonts/licplate-fonts-intro.html

[21] A. Vaishnav and M. Mandot, "An integrated automatic number plate recognition for recognizing multi language fonts," in *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE, 2018, pp. 551–556.

[22] M. J. Rahman, S. Beauchemin, and M. Bauer, "License plate detection and recognition: An empirical study," in *Science and Information Conference*. Springer, 2019, pp. 339–349.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[24] J. Brownlee. (2019) A gentle introduction to object recognition with deep learning. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/

[25] Y. LeCun *et al.*, "Generalization and network design strategies. connectionism in perspective," *Zurich, Switzerland, Elsiever*, 1989.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[27] L. Fei-Fei, A. Karpathy, and J. Johnson. Cs231n: Convolutional neural networks for visual recognition. [Online]. Available: http://cs231n.stanford.edu/index.html

[28] J. Brownlee. (2019) A gentle introduction to the rectified linear unit (relu). [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[29] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[30] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[31] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[32] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[34] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[35] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[37] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: http://arxiv.org/abs/1611.10012

[38] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," *CoRR*, vol. abs/1605.06409, 2016. [Online]. Available: http://arxiv.org/abs/1605.06409

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[41] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[44] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.

[45] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, “Cspnet: A new backbone that can enhance learning capability of cnn,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 390–391.

[46] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *Lecture Notes in Computer Science*, p. 346–361, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10578-9_23

[47] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” *CoRR*, vol. abs/1803.01534, 2018. [Online]. Available: http://arxiv.org/abs/1803.01534

[48] Tzutalin, “Labelimg,” https://github.com/tzutalin/labelImg, 2015.

[49] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: http://arxiv.org/abs/1612.03144

[50] D. Oreski, S. Oreski, and B. Klicek, “Effects of dataset characteristics on the performance of feature selection techniques,” *Applied Soft Computing*, vol. 52, pp. 109–119, 2017.