ROBUST REAL-TIME TRAFFIC SIGN AND LIGHT DETECTION WITH IMPROVED YOLOV7

A Thesis

by

YOUHAN SONG

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Gholamreza Langari
Committee Members,	Richard Malak
	Dezhen Song
Head of Department,	Guillermo Aguilar

August 2023

Major Subject: Mechanical Engineering

ABSTRACT

In this project, we seek to provide a computer vision model that can detect and classify traffic signs and lights for autonomous vehicles. Such a model enables proper information transmission to the controller so that suitable longitudinal/lateral actions can be taken based on the traffic conditions. To achieve a high-accuracy model, a significant amount of high-resolution and clear image datasets are usually necessary for training. However, such requirements might be hard to be satisfied, because the resolution of the image is determined by the resolution of the camera lens, weather conditions, etc.

Thus, the main objective of this paper is to propose an improved YOLOv7 model that has better performance than the original one in the real-time while the model is trained with random-quality and a limited number of datasets. At the end, we discuss the limitations of the proposed methods after applying and evaluating them with both images and videos of real-world driving by using the recorded rosbag. The evaluation is done with precision, recall, F-1 score, and mAP@0.5 for both the original and modified models and these values are used for comparison to identify the improvement. Finally, the conclusion is drawn based on the data comparison and the real-time detection results. We conclude that the modified model has a better performance than the original model for real-time autonomous driving.

DEDICATION

To my mother, father, grandfather, and grandmother who support and encourage me to complete the research.

To our friends who give advice when I face a problem while doing research and writing a paper.

To our teachers who helped to finish the research successfully and instructed us on the right track.

ACKNOWLEDGMENTS

I would like to acknowledge and want to say thanks to Dr. Reza Langari who helped me to complete the research successfully. His guidance and advice helped me to improve my research skill and writing skill. I would like to thank my committee members for participating in my defense and giving comments and suggestions for my project.

I would like to acknowledge and want to say thanks to my lab members. Their assistance was helpful for me to solve the problem, collect the data, and complete my project.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Richard Malak of the Department of Mechanical Engineering and Professor Dezhen Song of the Department of Computer Science & Engineering.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a teaching assistantship from the Department of Mechanical Engineering, Texas A&M University, and the thesis research assistantship from the same foundation.

NOMENCLATURE

Algorithm	Procedure used for solving problem or performing a computation
DL	Deep learning
CNN	Convolution Neural Network
High Speed	more than 65 mph
Moderate Speed	around 35 mph
GPS	Global Positioning System
IMU	Inertial Measurement Unit
GPU	Graphics Processing Unit
R-CNN	Region proposal with Convolution Neural Network
SSD	Single-shot multi-box detection
YOLO	You Only Look Once
mAP	mean Average Precision
ROS	Robot Operating System
LabelImg	Labeling took for image
Rosbag	the recorded data through ROS
Roboflow	the webstie that has a massive dataset

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

There are many challenging tasks in sensing for autonomous driving. Specifically, Object detection and classification are among the ones with significant tasks among them. Based on the information, the controller plans and decides the motions and paths for the vehicle to take so that it can safely reach the desired destination. Therefore, a computer vision model that has a high accuracy for detecting and classifying objects is needed. However, building such a model is not trivial due to the requirements of the training dataset, e.g., data quality and the amount of available data. Moreover, the model architecture is made of a deep learning (DL) network, i.e., Convolution Neural Network (CNN) [5], whose layer combination decides the accuracy of the model and thus needs a proper design.

The sensors to detect objects consist of a camera, lidar, and radar. Among them, the camera is commonly used for object detection because it is relatively cheaper, easier to install, and has higher accuracy of detection than others. Therefore, a computer vision model based on the camera is considered in this work and one of the most popular algorithms is the YOLO (You Only Look Once). Among series, the YOLOv7 model has been chosen to use after the comparison of other series in terms of accuracy, detecting speed, loss, etc.

When driving in the real world, unexpected situations might happen. For instance, one may encounter construction areas, restricted road usage (e.g., carpool lanes), or emergency situations (e.g., traffic accidents). Most human drivers recognize the road situation by looking at the traffic signs and slowing down the vehicle when they face these situations, while autonomous vehicles may need to rely on the perception system, motion planner, and controllers to accomplish these tasks. Most computer vision models that have already been developed and published can detect the common object on the road including cars, trucks, stop signs, etc. However, these models do not have the function to detect and classify traffic signs and lights since they are not trained to do so. Furthermore, DL heavily relies on the quality and amount of data for traffic signs and lights and those features are similar. Consequently, the computer vision model might be confused such that

it would detect and classify traffic signs with low accuracy, which may lead to the improper decision of the motion and path planning for the autonomous vehicle. Moreover, since the computer vision model should detect in real-time with high vehicle speed, it requires high accuracy and fast detection at the same time.

Due to the time requirements of collecting and annotating the dataset, and the difficult knowledge of CNN, it is challenging to make a computer vision model that requires massive amounts of the dataset and would be able to detect the object with high accuracy. In this study, we are going to address the problem by explaining the current method and proposing a reinforced model based on the existing model. In this paper, we will first do a literature review to explain the background of the problem being considered. Then, we will discuss problem identification, methodology, implementation notes, assumptions, and limitations. Lastly, we will present the results and finish with the concluding remarks.

2. LITERATURE REVIEW

2.1 Evolution of Real-Time Object Detector

In the early 2000s, object detection technology was developed based on handcrafted features, and this technology is called traditional detectors [6]. Due to the lack of skills to extract the features of the images effectively, the developer should design the feature extractor complicated. These detectors are included in traditional detectors: Viola Jones Detectors [7, 8], HOG Detector [9], and Deformable Part-based Model (DPM) [10, 11, 12]

In 2012, since convolution neural networks (CNN) have been developed [5], it is possible to train a CNN to extract the unique and specific features of an image. There are two different genres for object detection which are one-stage detection and two-stage detection [13]. For the one-stage detection, when image classification and segmentation are done, they use a regular and dense sample that consists of locations, scales, and aspect ratios. The You Only Look Once (YOLO) series [4, 14, 15, 16, 1, 17], Single-shot multi-box detection (SSD) [18], RetinaNet [19], and RefineDet [20] are typical examples of one-stage detection. Figure 2.1 shows the architecture of the YOLO.

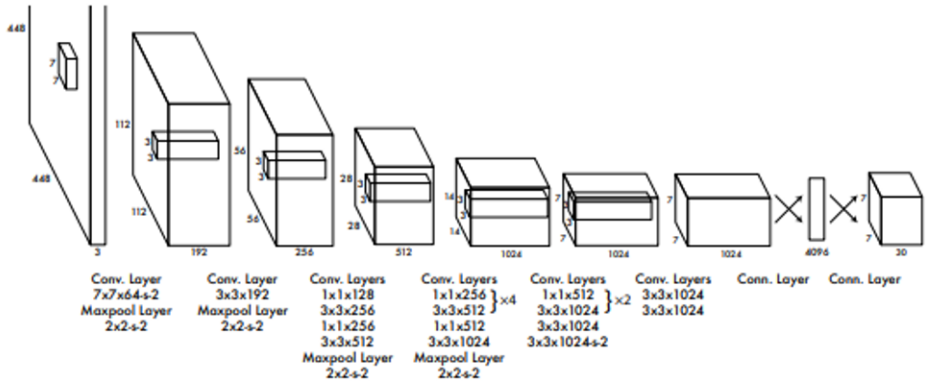


Figure 2.1: Architecture of YOLO [1]

For two-stage detection, the region or object proposals will be done in the first step, and based on the proposals, they will be classified and captured by bounding boxes in the second step. Region proposal with convolution neural network (R-CNN) [2], spatial pyramid pooling (SPP) [21, 22], and Feature pyramid network (FPN) [23] are typical examples of two-stage detection. Figure 2.2 shows the architecture of the R-CNN.

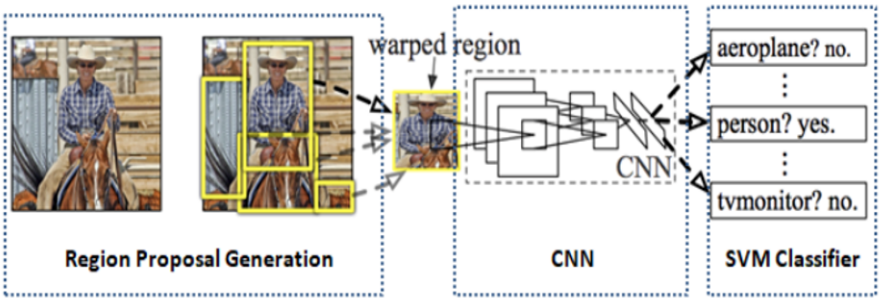


Figure 2.2: Architecture of R-CNN [2]

By comparing the one-stage detection and two-stage detection, Figure 2.3 [3] shows the speed and the accuracy depending on different models.

Model architectures		Image size (pixels)	mAP@0.50 (%)	mAP@0.50:0.95 (%)	Inference time (ms/image)	# Model parameters (million)
One-stage	Yolov5n	640×640	76.58 (1.33)	55.34 (1.58)	**17** (3)	**1.8**
	Yolov5s	640×640	77.47 (1.10)	58.41 (3.08)	50 (5)	7.3
	Yolov5m	640×640	78.64 (0.93)	60.89 (1.02)	132 (11)	21.4
	EfficientDet D0	512×512	75.23 (0.97)	55.94 (1.41)	173 (9)	3.9
	EfficientDet D1	640×640	76.55 (1.24)	57.76 (0.92)	176 (13)	6.6
	EfficientDet D2	768×768	77.83 (2.41)	58.04 (0.83)	182 (5)	8.1
	RetinaNet R50-FPN	512×512	78.50 (2.13)	60.20 (2.40)	115 (12)	37.9
	RetinaNet R101-FPN	512×512	**79.98** (1.18)	**62.97** (0.98)	118 (10)	56.8
Two-stage	Fast RCNN R50-FPN	512×512	75.36 (0.92)	57.75 (1.75)	175 (8)	27.4
	Faster RCNN R50 - C4	512×512	75.05 (1.66)	55.02 (1.26)	196 (5)	33.8
	Faster RCNN R101 - C4	512×512	75.47 (0.96)	58.96 (2.20)	256 (13)	52.7
	Faster RCNN R101-FPN	512×512	76.92 (2.66)	58.73 (2.20)	229 (9)	60.6
	Faster RCNN X101 - FPN	512×512	78.37 (1.61)	61.48 (1.77)	277 (11)	104.8

Figure 2.3: Comparison of Models [3]

Figure 2.3 indicates that the one-stage method in general has high speed and less accuracy. On the other hand, the two-stage method overall has low speed and high accuracy. However, the result from the figure says that the accuracy of the one-stage method and the two-stage method is similar, whereas the one-stage method especially YOLO5n is faster than the two-stage method. Therefore, the one-stage method is suitable for real-time detection because the system needs to detect the object as fast as it can.

2.2 Object Detection for Autonomous Driving

Over the last few decades, as the area of object detection and classification for autonomous vehicles using computer vision has been studied and developed, there is plenty of literature to discuss. Many studies mainly discuss low-speed driving in the urban area instead of high-speed driving on the highway regarding the obstacle detection and classification. To the author's best knowledge, no published literature addresses the work in high speed and traffic signs and lights at the same time. Since there is no work related to this, the dataset of traffic signs and lights does not exist such that it needs to be collected and annotated.

Reference [24] is one of the papers for object detection in high speed, studied by IIITDM Kancheepuram, Chennai, India, in 2017. In this article, they use the Faster Region proposal with convolution neural network (R-CNN) [25] which is a two-stage detection algorithm [13] to detect the object in the urban and highway, and it is implemented in GPU. This model used ZF net which is pretrained in ImageNet and tuned in PASCAL VOC 2012 dataset which includes 20 object classes such as bicycle, sofa, TV monitor, boat, airplane, bus, cow, car, bottle, chair, sheep, dining table, bird, cat, person, dog, potted plant, horse, train, and motorbike [26]. Among them, they mainly deal with objects on the road such as bus, car, motorbike, and person. The result has been described successfully, but this method does not detect the traffic signs and lights because its dataset does not have any. And this model is not proper for this work, because the two-stage detection algorithm is more focused on accuracy rather than speed.

Reference [27] is the paper that addresses the result of traffic sign detection and classification at moderate speed (up to 35 mph), completed by University Tun Hussein Onn, Malaysia, in 2020. In

the study, they proposed the YOLOv3 [15] model which is a one-stage detection algorithm [13] to detect traffic signs, and it is implemented in both CPU and GPU. The class of the dataset includes stop, no entry, parking, speed limit, bump, and traffic light. This paper could be a good example of how the number of images plays an important role in accuracy. They explain the relationship between accuracy and the number of collected datasets by showing the graph and it denotes that the accuracy is higher as the number of images is plenty. Therefore, in our work, we will collect more datasets than this, and expect that our work will have better performance than this. Moreover, only GPU will be used to train the model such that we believe this will play a role in improving the performance as well.

Reference [28] gives good motivation although it is for object detection for the drone. In their work, they point out the disadvantages of the activation function of original yolov5 and propose changing them such that they could increase the performance as well as reduce the training time. And they compare the improved model to the other Yolo versions which are YOLOv3 and YOLOv4. It is worth testing this method in our work because it works in YOLOv5, and YOLOv7 has a similar architecture so we can expect that this might improve the performance of it.

Reference [4] is the paper that explains the YOLOv7, developed by the Institute of Information Science, Academia Sinica, Taiwan. This explains the architecture of YOLOv7 and compares YOLOv7 and other versions in terms of the inference time (speed) and accuracy (average precision). In this paper, it shows that YOLOv7 is faster and more accurate than other YOLO series. From this, this model is considered to be used for this project. A detailed discussion will be described in the methodology.

3. SIGNIFICANCE

Testing and verifying that robust real-time object detection algorithm is working on the running vehicle is important before it is utilized on the actual road such as the freeway and highway. Moreover, the dataset consists of traffic sign and light information that would be collected, annotated, and published when the code is implemented.

Other project – [24], [27], and [28]- consider a similar topic: Evaluating and analyzing the performance of YOLO trained for their specific purposes. Although the implementation of the algorithm is similar, [24] and [28] mainly talk about the object which could be commonly detected on the urban road. While [27] does the traffic sign detection, the dataset is not annotated specifically. In our work, the dataset will be annotated in detail by labeling the type of signs. This would be more helpful for an autonomous system to transmit the information and decide what action should be taken.

4. PROBLEM IDENTIFICATION

On the road, especially on the highway and freeway, there are several kinds of traffic signs that cameras could detect such as speed limit, stop signs, warning signs, etc. However, since the highway is wide and long, and the vehicle is moving at high speed, it is hard for the camera installed in the vehicle to capture traffic signs. Moreover, the quality and number of images for traffic signs and lights are probably unsure to augment the accuracy. Our goal is to choose the best computer vision model for real-time detection and improve it by modifying the parameter which could potentially upgrade the performance.

Sometimes, traffic signs might get damaged by factors such as weather, accident, etc., and it is hard to repair and maintain the traffic sign. Furthermore, since features of the traffic sign and light are similar in terms of shape and color, it would be hard for computer vision model to predict the object which could increase the chance of occurrence of an accident.

From this, we need to develop a computer vision model that could extract the detailed features of the image for training as well as be able to capture them from the given information for testing.

To accomplish this, we considered and tested several object detection algorithms which are trained with massive datasets which is the MS COCO dataset (i.e., YOLO series). However, in this dataset, there are no traffic signs and lights other than stop signs such that models cannot detect and classify. Therefore, we decided to train the modified model with collected data from the internet and pictures taken by our team.

Despite these limitations and difficulties, we expect that the real-time object detection algorithm which will be chosen after testing, possibly overcome these problems by modifying the parameter. In detail, we are going to propose a method which is replacing the activation function at the end of each convolution block. This might give an influence on the overall calculation while both training and testing. It is because output will be minimized, maximized, or vanished when the input (the value that image is calculated through convolution and batch normalization) passes the activation function. Due to the uncertainty of how much data is enough for the training, the

algorithm should be tested and needs to be upgraded to successfully achieve the desirable performance depending on the usage. These ideas might have better performance for the detection and classification of traffic signs in case the vision of the camera is blurred – for instance, the camera's lens may be out of focus which makes the computer vision model not be able to extract the proper information from the captured image. Furthermore, the model would be able to detect and classify the object from the image in bad weather conditions such as rain, snow, fog, etc. As mentioned above, there is the uncertainty of sufficient datasets for traffic signs and lights. It may not be possible to recognize and fail to detect the target as a result.

5. METHOD

5.1 System Design

5.1.1 Original Model

Due to the usage of object detection, the best model needs to be chosen based on speed and accuracy. As noted above, the one-stage method is proper to use in real-time object detection, and we decided to choose one of the models in the YOLO series. YOLO has been developed gradually, and there are 7 typical YOLO versions until 2023. They are considered, and the comparison is shown in Figure 5.1.

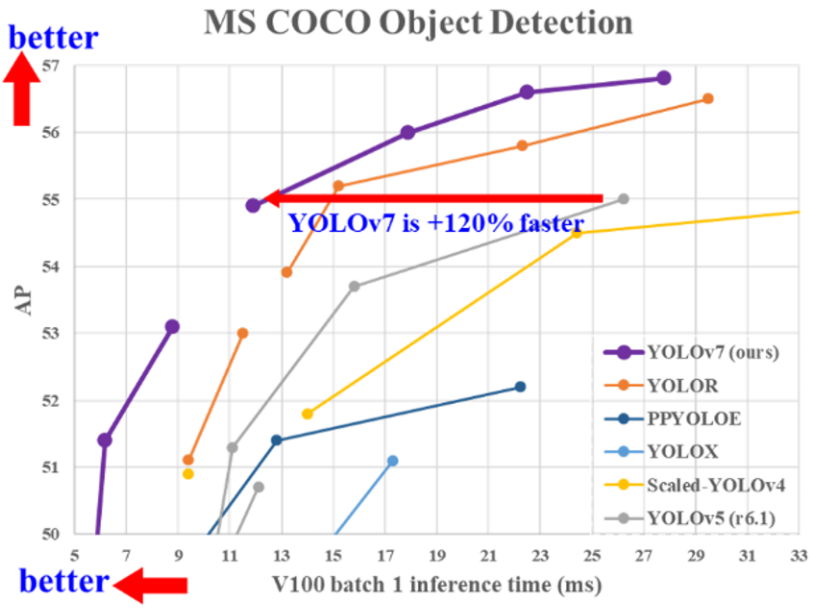


Figure 5.1: Comparison of YOLO series [4]

As the graph is located in the top left, it means that the model's speed is fast. The figure indicates that the YOLOv7[4] model is the fastest and most accurate model among real-time object detectors. The structure of the YOLOv7 is consistent with three pipelines which are the backbone,

neck, and head, and Figure 5.2 shows the architecture of the YOLOv7.

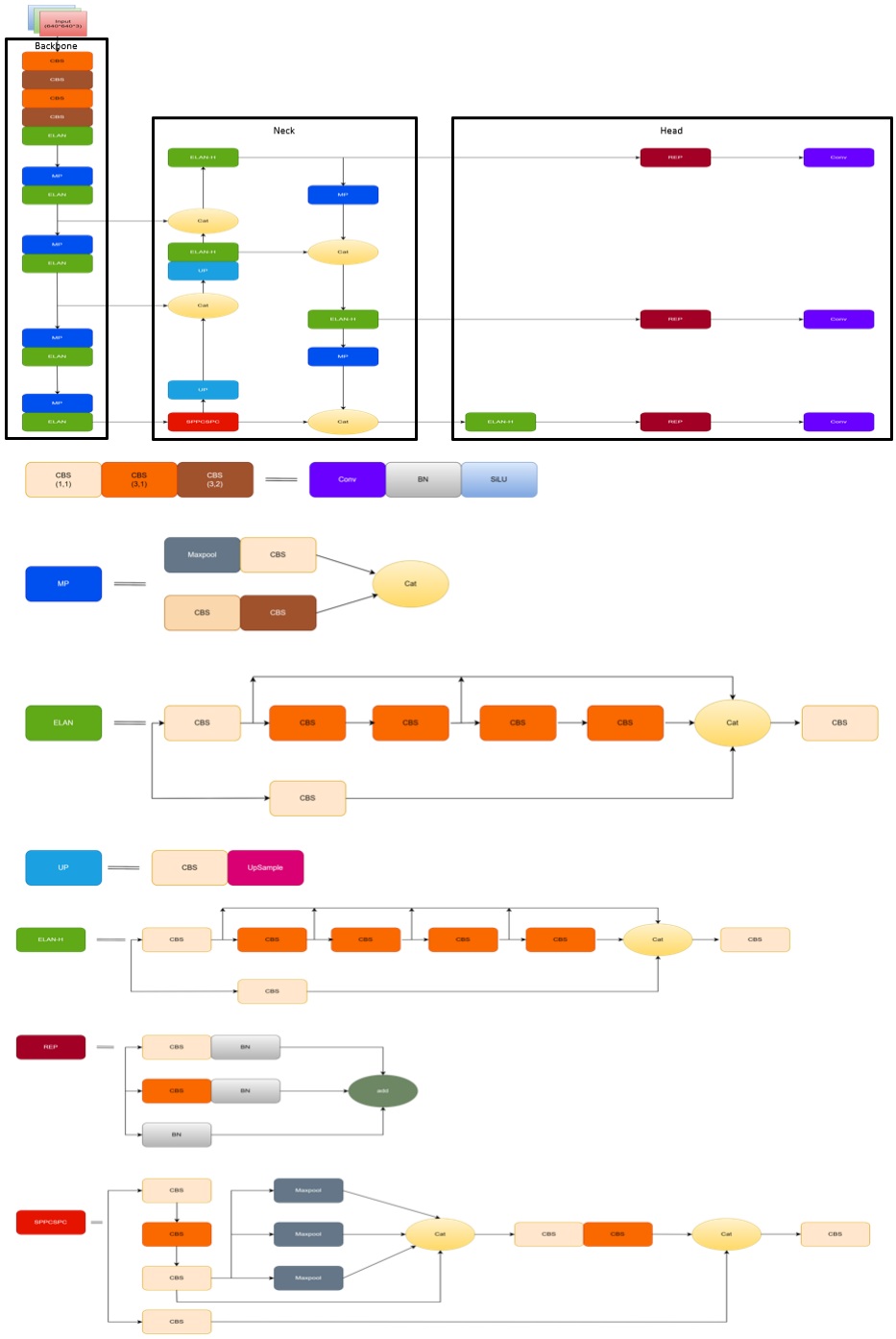


Figure 5.2: Architecture of YOLOv7

Backbone is the convolution neural network (CNN) that extracts, calculates, and augments the features of the image and sends it to the Neck. To extract the feature efficiently, the structure of the backbone consists of a group of convolution blocks, a maxpooling block, and an Efficient Layer Aggregation Network (ELAN) [29] block. The ELAN block has two ways: the first way has 5 convolution blocks, and the second way has only one convolution block. In the end, the information from the first, third, and fifth from the first way and the one from the second way are concatenated at the end and it goes through another convolution block to downsize the channel. This would make the model possibly reinforce the extracted features. The neck part receives features from the backbone and strengthens the extracted features. To this, Neck has a PAFPN bottom-up path aggregation structure [30] and its structure has an SPPCSPC module, upsampling module, and ELAN-H module. The difference between the ELAN-H module and the ELAN module, the ELAN-H module brings information from every term of the convolution block, and its information is concatenated to the second way and downsized by passing the convolution block. By doing this, it could prevent the losing gradient during convolving layers. Finally, the head part receives the input from the neck, and it outputs detection results such as probabilities of classes, confidence scores, and bounding boxes.

5.1.2 Our Model

In the original version, SiLU (Sigmoid-weighted Linear Unit) activation function is used in the convolution block. This is because compared to the other activation function, this could prevent vanishing gradients while training which is known as dying ReLU (Rectified Linear Unit). Because the function will produce the output as 0 when x (input) is negative (the input is the value after the computation of convolution and batch normalization in the convolution block). If this happens, it will drop the accuracy of detecting and classification. Equations are stated below, and Figure 5.3 shows the graph of RELU and its derivative, and Figure 5.4 shows the graph of SiLU and its derivative.

$$RELU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} \tag{5.1}$$

$$RELU'(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases} \tag{5.2}$$

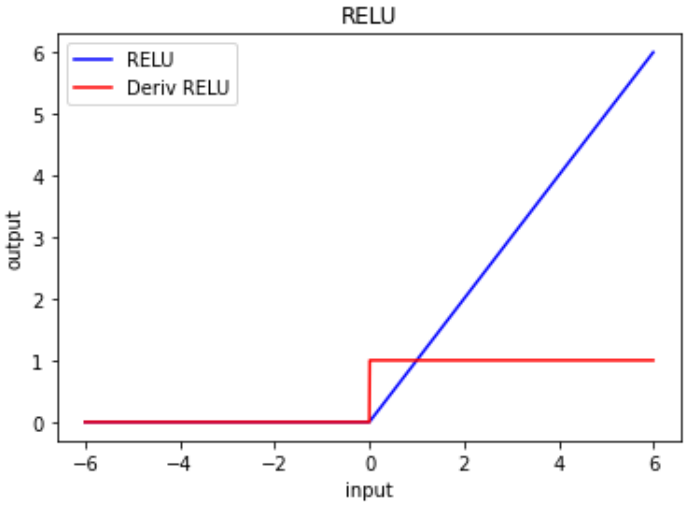


Figure 5.3: Graph of RELU and derivative of RELU

$$SiLU(x) = x * \frac{1}{1 + e^{-x}} \tag{5.3}$$

$$SiLU'(x) = \frac{1}{1 + e^{-x}} + (x * \frac{1}{1 + e^{-x}}) * (1 - \frac{1}{1 + e^{-x}}) \tag{5.4}$$

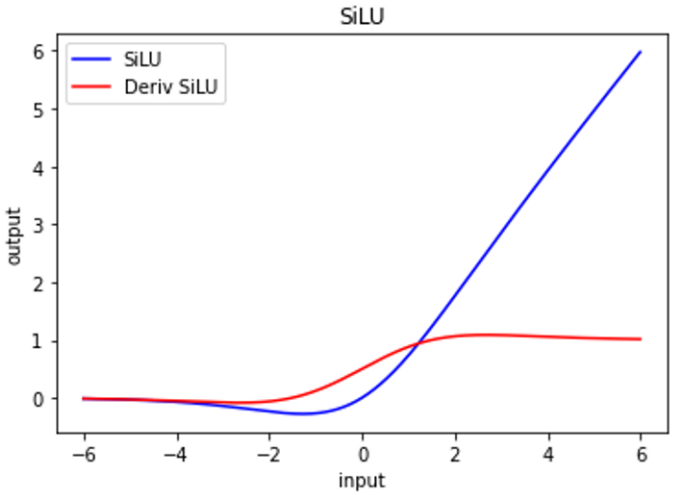


Figure 5.4: Graph of SiLU and derivative of SiLU

However, similar to the ReLU activation function, this graph shows that it also has a limitation that when x is close to the negative infinite, the output is close to 0 if the result of convolution is negative. To solve this issue, we decided to use ELU (Exponential Linear Unit) instead of SiLU for the activation function. The advantages are that this could reduce the time for training as well as improve the performance of the testing of the neural network. This is because ELU has a negative value, and this will push the mean close to 0, and the bias will be reduced. Therefore, this will speed up the training by shifting the normal gradient closer to the unit natural gradient. Moreover, unlike ReLU and SiLU, this could produce negative outputs, thus this would be able to remember and transfer gradients albeit it goes through many computations of convolution. On the other hand, since the graph shows that when the input is greater than 0, and the range of output is [0, ∞], this might cause the blowing up of the activation. Equations are stated below and the value of the α is 1 (otherwise this function will be changed to a different one), and Figure 5.5 shows the graph of ELU and its derivative.

$$ELU(x) = \begin{cases} \alpha e^{-x} - 1, & x \leq 0 \\ x, & x > 0 \end{cases} \tag{5.5}$$

$$ELU'(x) = \begin{cases} -\alpha e^{-x}, & x \leq 0 \\ 1, & x > 0 \end{cases} \tag{5.6}$$

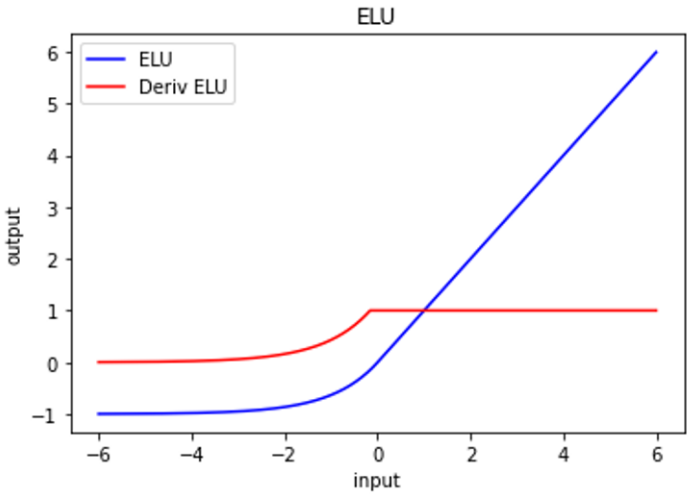


Figure 5.5: Graph of ELU and derivative of ELU

As the activation function for the convolution block has been changed, the whole block has also been modified. Because most of the blocks consist of convolution blocks. In detail, CBS consists of convolution, batch normalization, and activation function, and ELAN, MP, ELAN-H, SPPCSPC, and REP contains CBS. This means that this will have an influence on the calculation. Figure 5.6 shows the overall architecture of modified YOLOv7 to help with understanding.

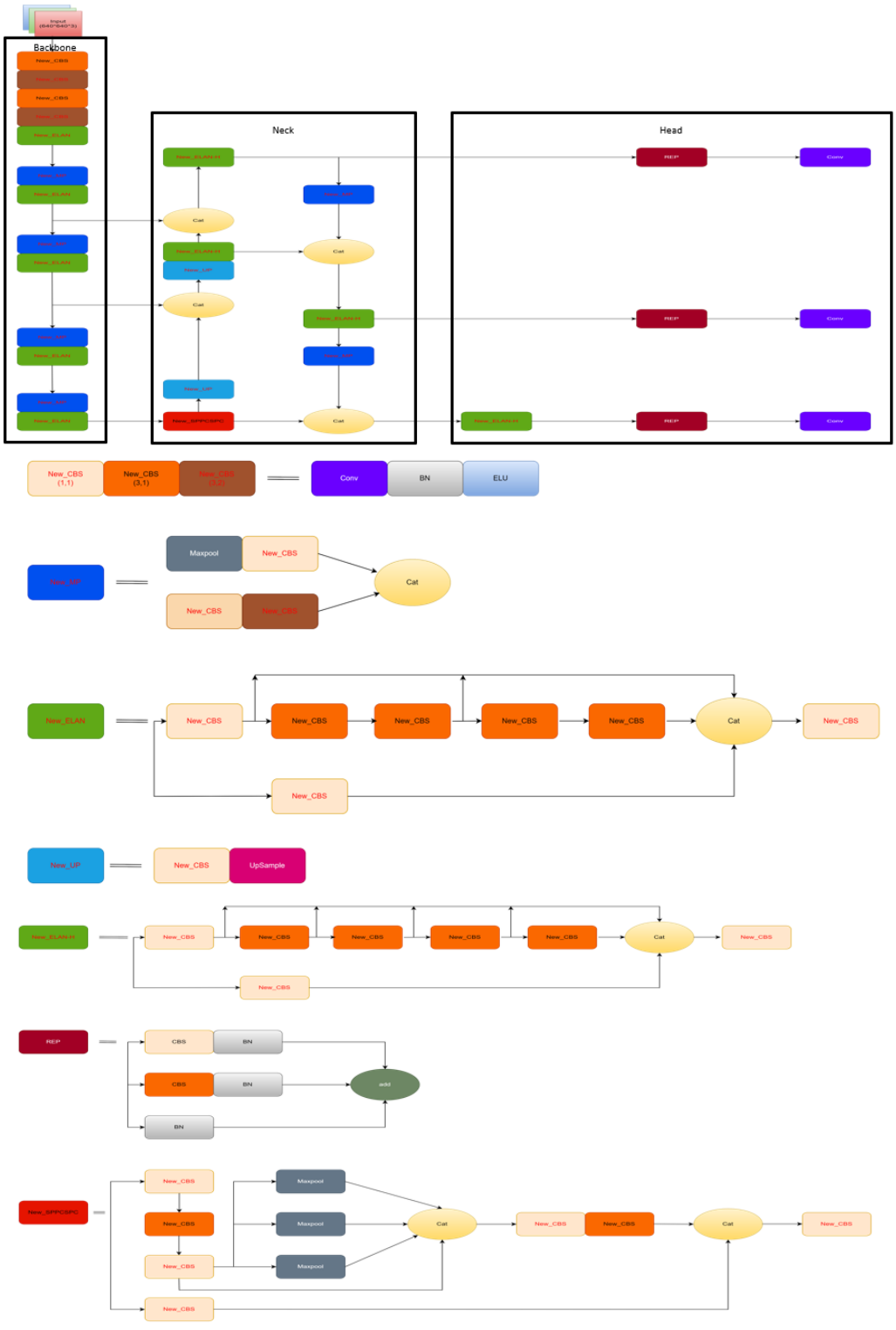


Figure 5.6: The architecture of modified YOLOv7

5.2 Strategies

The objective of this work is to modify and improve the detecting and classifying accuracy and speed of YOLOv7 which could detect and classify traffic signs and lights. YOLOv7 will be trained with 14 classes which are speed limits: 30, 35, 40, 45, 50, 55, 60, 65, 70, and 75 mph, traffic lights: green, yellow, red, and stop signs. Table 5.1 below shows classes, and the number of images per class.

Table 5.1: Number of Dataset

classes	Number of images
Speedlimit: 30 mph	540
Speedlimit: 35 mph	1636
Speedlimit: 40 mph	1741
Speedlimit: 45 mph	1724
Speedlimit: 50 mph	1789
Speedlimit: 55 mph	2131
Speedlimit: 60 mph	1675
Speedlimit: 65 mph	1834
Speedlimit: 70 mph	1838
Speedlimit: 75 mph	1739
Stop	2194
Green	6901
Yellow	1371
Red	4892

To train models, the total dataset which has 29978 images needs to be distributed to training,

validation, and testing set, and the testing set is 20 images chosen randomly and 85% of the rest of the data is set as a training set and 15% of the data is set as a validation set. To maximize the number of datasets for the training, only 20 images are decided to be used to test each class. Therefore, the number of training, validation, and testing datasets is 25481, 4477, and 20, respectively. After distribution, the model is going to be trained in these hyper-parameters: 32 batch size, 640x640 image (pixel) size, 300 epochs, and 0.01 learning rate.

After training is done, the model is going to be run with the recorded rosbag for testing. The rosbag is not used to extract the dataset and is not used for training. To do this, the Python code is written to make nodes communicate. Nodes consist of publisher and subscriber. The publisher release and send the information, and the subscriber receives its information and calculates the result using its algorithm. Apply this concept to the work, the images of the camera in the rosbag (publisher) are transmitted to the Python code (subscriber) which will generate and apply the weight of the model and display the result of the detection and classification.

5.3 Data Collection

The dataset of traffic signs and lights are collected which are currently used in the United States of America because their design of them slightly varies from country to country. The data is the raw image collected in the city and highway in Texas. Moreover, some images are obtained through the internet such as roboflow for various types of datasets as well as gathering as much data as possible. Additionally, the stop sign images are extracted from the MS COCO dataset and used since the dataset contains plenty of images for the stop sign and its design is similar. To do this, we brought and modified the Python code to extract the Stop sign from the dataset. After collecting the whole data, the Python code that converts the class is developed to replace the original class number with our class number. Finally, we put the total images and labels in the image and label folder together for the training. The data for simulation is recorded in a rosbag to test the accuracy and detection of the YOLO.

5.4 Labeling Tool

Before images are used to train, they are supposed to be annotated, and we used LabelImg. Through the terminal in Ubuntu, we could open the tool, and Figure 5.7 and Figure 5.8 show the command and what the tool looks like. Depending on the number of traffic signs and lights, there might be multiple bounding boxes that are supposed to be drawn and labeled. While the YOLO is trained, it would predict the class and number of objects with many anchor boxes by calculating and finding out the highest intersection over Union (IOU).

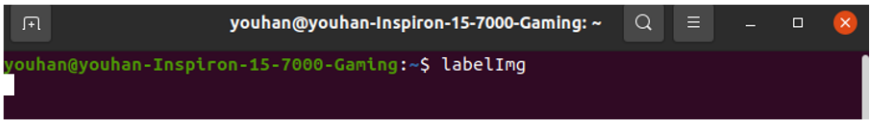


Figure 5.7: Command to open labelImg

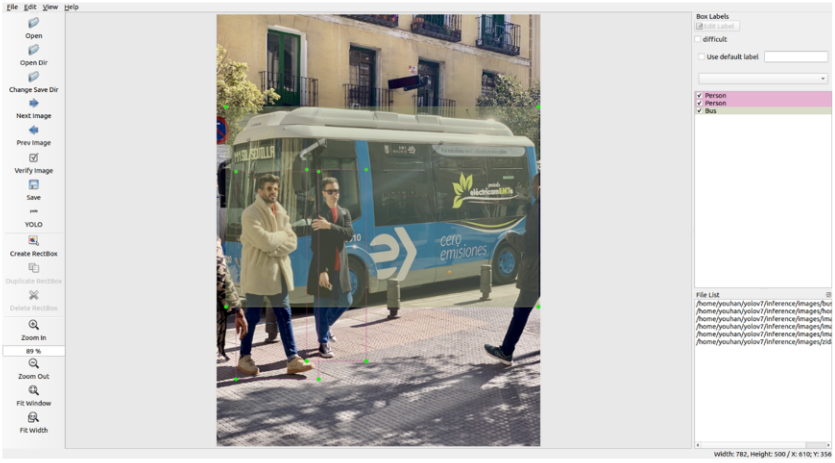


Figure 5.8: Example of how to label using labelImg

6. IMPLEMENTATION NOTES

The YOLOv7 is trained in Ubuntu 20.04 LTS. The specific component of YOLOv7 code is implemented in Python and its structure and hyperparameters are described in yaml format. The specifications for the computer which are used to train are that AMD® Ryzen Threadripper 39060x 24-core processor x48 and two graphic processors which are both Nvidia GeForce RTX 3090 TI.

The simulation using YOLOv7 is implemented in ROS 1 (melodic) on Ubuntu 18.04 LTS. The detecting algorithm with the trained model is implemented in Python and the video for the simulation is recorded as a rosbag file. The specification for the computer which is used for the simulation is that Intel® Xeon® Silver 4114 CPU 2.20GHZ x 20 and a graph processor of Nvidia GeForce RTX 2080 TI.

The memory usage of the graphic processor will be listed below, and it will address the capabilities of GPU memory of each computer based on the purpose of the usage.

To help to understand, 1 MiB (Mebibyte) = 1024 Kb (kilobyte) and 1 MiB = 1.04858 MB (Megabyte).

For training Yolov7, the approximate total available memory is 48GB (45776.4 MiB) with two Nvidia GPUs (3090 TI). For the original Yolov7, it costs 20.2 - 21.2 GB to train the model. For the modified YOLOv7, it costs 20.9-23.1 GB to train the model.

For testing trained models, the approximate total available memory is 11 GB (10490.4 MiB) with an Nvidia GPU (2040 TI). The original YOLOv7 requires 1278 MiB (1.34 GB) to implement the Python code. The modified YOLOv7 model requires 1292 MiB (1.35 GB) to run the Python code. These requirements of memory include running the rosbag file at the same time.

Lastly, the investigation of the minimum number of annotated images per class to train the model is discussed and we assume that at least high hundreds of annotated images for each class are required, and it can be fulfilled with the open dataset, created by the public. However, we didn't record and extract the imagery from the test routes because this will cause overfitting while training, and this would misinterpret the result. There are many different object detection algorithms,

and their performance could be considered good when it is trained using images that are brought from the training dataset. Through this, we could estimate how well algorithms perform. However, it is hard to prove that the model would work if it were applied to the unlearned real-time image.

7. ASSUMPTIONS

Since there are many different variants while running real-time autonomous vehicles, we set a certain environment for this experiment. So, we assumed that the camera is running up to 30 Hz depending on the brightness or other factors. And the real-time sign and light detection model is running on the vehicle. Lastly, the vehicle is running on the actual road.

8. LIMITATIONS

During the experiment, we found out that there are limitations. First, a camera does not experience any noise from other factors such as weather (snow, rain, fog, etc.). Second, the uncertainty of how many data is required to develop and reinforce the accuracy of the model. Third, the resolution of images for the training dataset are various which means that some are blurred, another is noised, and other is clear. Lastly, the resolution might be changed depending on the speed of the vehicle. If the vehicle is running at high speed, the camera will have a lower resolution.

9. RESULTS

9.1 Performance Evaluation

9.1.1 Evaluation Method

The performance of the system will be evaluated for the accuracy of the detection and classification, and this will be proved with precision, recall, F-1 score, Average Precision (AP), and mean Average Precision (mAP). In this experiment, with the threshold at α, the true positive is considered only if the Intersection over Union (IoU) of ground truth and predicted bounding box is greater than α. A false negative is defined if IoU is less than α or if there is a misdetection.

$$\text{Intersection over union (IoU)} = \frac{A \cap B}{A \cup B} \tag{9.1}$$

$$\text{Precision} = \frac{\text{TP (True Positive)}}{\text{TP(True Positive)} + \text{FP (False Positive)}} \tag{9.2}$$

$$\text{Recall} = \frac{\text{TP (True Positive)}}{\text{TP (True Positive) + FN (False Negative)}} \tag{9.3}$$

$$\text{F1-Score} = \frac{\text{2*Precision*Recall}}{\text{Precision + Recall}} \tag{9.4}$$

Intersection over Union (IoU) is the area of overlap divided by the area of union. Precision is the ratio of how many corrected objects are detected in overall detected results. Recall is the ratio of how many true positive predicted detections is made in the overall ground truth. In other words, it indicates how well the model could detect the target. F1-score is the harmonic mean of precision and recall. The value of the F1-score is between 0 and, 1 and if its value is close to 1, it means that the accuracy of the detecting object is high, whereas if its value is close to 0, it means that the accuracy of the detecting object is low.

$$AP@\alpha = \int_0^1 p(r)dr \tag{9.5}$$

$$mAP@\alpha = \frac{1}{n}\sum_{i=1}^{n} AP_i \tag{9.6}$$

Mean Average Precision (mAP) is one of the most commonly and widely used to evaluate the accuracy of object detection and classification. Mean Average Precision (mAP) could be calculated by getting the Average Precision (AP) from each class and computing the average of all Average Precision. In the equation 9.5, p represents the score of the precision and r represents the score of the recall. In the equation 9.6, n denotes the number of classes.

9.1.2 Loss Function

The loss function is the function to check how well the model can predict the output by comparing the actual target while training. To make a better model, the goal is to minimize the loss between the actual target and predicted output. For this, there are three losses that consist of total loss function which are Regression Loss, Confidence Loss, and Classification Loss. The BCE With Logits Loss (cross-entropy loss) is used for the Confidence and Classification Loss, and cIoU (Complete-IoU) loss is used for the regression loss.

Confidence loss is to check whether there are targets in the predicted bounding box. The equation is stated below.

Confidence loss:

$$\mathcal{L}_{obj}(P_p, P_{IoU}) = BCE^{sig}_{obj}(P_o, P_{IoU}; \omega_{obj}) \tag{9.7}$$

In the equation 9.7, BCE^{sig}_{obj} stands for the loss of binary cross-entropy, P_0 stands for the confidence score of predicting objects in the frame, and P_{IoU} stands for the IoU score of the corresponding object in the prediction frame and only ground-truth is used to calculate, and ω_{obj} stands for the weights of a positive sample.

Classification Loss is the function to check whether the model can predict the discrete class and label outputs. The equation is stated below.

Classification Loss:

$$\mathcal{L}_{cls}(C_p, C_{gt}) = BCE_{obj}^{sig}(C_p, C_{gt}; \omega_{cls}) \tag{9.8}$$

In the equation 9.8, BCE_{obj}^{sig} represents the loss of binary cross-entropy, C_p represents the predicted value of corresponding classes, C_{gt} represents the ground truth value of corresponding classes, and ω_{cls} represents the weights of a positive sample.

Regression Loss is to measure how well the model can predict the bounding box compared to the ground truth bounding box. The equation for Regression Loss is stated below.

Regression Loss:

$$\mathcal{L}_{cIoU} = 1 - IoU + \frac{\rho^2(b^p, b^{gt})}{C^2} + \beta v \tag{9.9}$$

$$\beta = \frac{v}{1 - IoU + v} \tag{9.10}$$

$$v = \frac{4}{\pi^2}(\arctan\frac{\omega^{gt}}{h^{gt}} - \arctan\frac{\omega^p}{h^p})^2 \tag{9.11}$$

$$\rho^2(b^p, b^{gt}) = ||b^p - b^{gt}||_2 \tag{9.12}$$

In the equation 9.9, 9.10, 9.11, and 9.12, b^p denotes the prediction of the bounding box, b^{gt} denotes the ground truth of the bounding box, ρ denotes the distance between the center point of the prediction bounding box and the center point of the ground truth bounding box, C denotes the maximum boundary distance between the predicted bounding box and the ground truth of the bounding box, ω^{gt} denotes the width of the ground truth of the bounding box, ω^p denotes the width of the prediction of the bounding box, h^{gt} denotes the height of the ground truth of the bounding

box, and h^p denotes the height of the prediction of the bounding box.

The total loss is the sum of these three losses, and it can be expressed below.

Total Loss:

$$\mathcal{L}_{total} = \sum_{k=0}^{K}[\alpha_k^{balance}\alpha_{box}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\Gamma_{kij}^{obj}\mathcal{L}_{cIoU} + \alpha_{obj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\Gamma_{kij}^{obj}\mathcal{L}_{obj} + \alpha_{cls}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\Gamma_{kij}^{obj}\mathcal{L}_{cls}] \quad (9.13)$$

In the equation 9.13, K indicates the output of the feature map, S^2 indicates the number of cells, and B indicates the number of anchors on the cells. $\alpha_k^{balance}$ indicates the output of the kth feature map at each scale, which is 80x80, 40x40, and 20x20, and in the experiment, the values are 4.0, 1.0, and 0.4, respectively. α_{box}, α_{obj}, and α_{cls} indicate the loss gain of bounding box, confidence, and classification, and these are 0.05, 0.7, and 0.3, respectively. Γ_{kij}^{obj} indicates whether the object is in the jth bounding box in ith cell in the kth feature map. If the object exists, the value is 1. Otherwise, the value is 0.

9.1.3 Evaluation Result

As mentioned in the method, both the original and modified models are trained at 300 epochs, 32 batch sizes, 640x640 image (pixel) size, and 0.01 learning rate. It takes about 26 hours to train for both original and modified model. The result of object detection for the original and modified models are shown in Figure 9.1 and Figure 9.2. In the figures, graphs of Precision, Recall, and F-1 score are shown. Additionally, to compare the result objectively, these are evaluated with values. Table 9.1 shows the summary of Precision, Recall, F-1 score, and mAP at 0.5. For the original model, it has 96.1% for precision, 96% for recall, 89% for F1-score, and 92.3% for mAP@0.5.

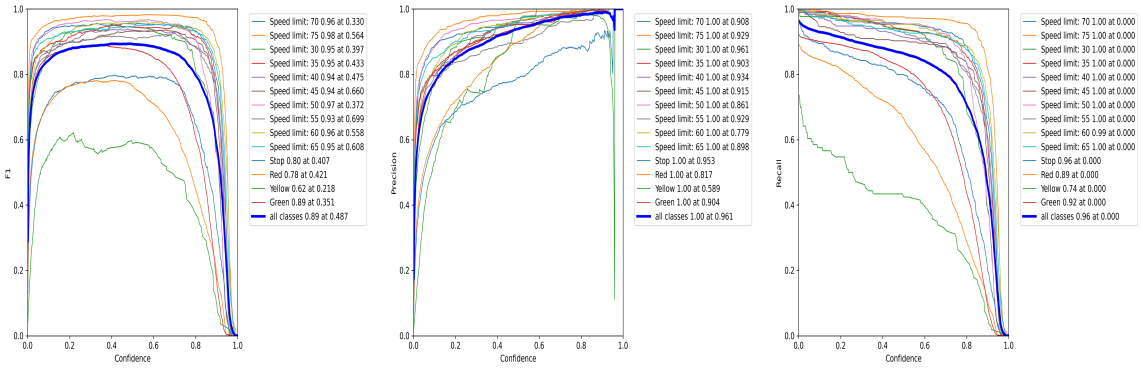


Figure 9.1: Result of Original model

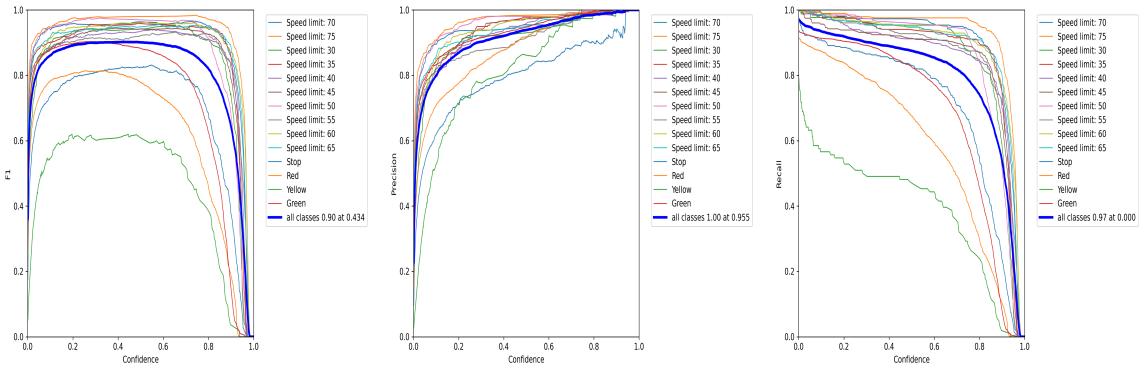


Figure 9.2: Result of Modified model

Table 9.1: Comparison of performance

Model	Precision(%)	Recall(%)	F-1 score(%)	mAP@0.5(%)
Original YOLOv7	96.1	96	89	92.3
Modified YOLOv7	95.5	97	90	93.4

By evaluating and comparing the result, we found that our modified model has a better perfor-

mance than the original model. For the modified model, it has 95% for precision, 97% for recall, 90% for F1-score, and 93.4% for mAP@0.5. Figure 9.3 and Figure 9.4 show the graph of loss, and Precision-Recall during the training process.

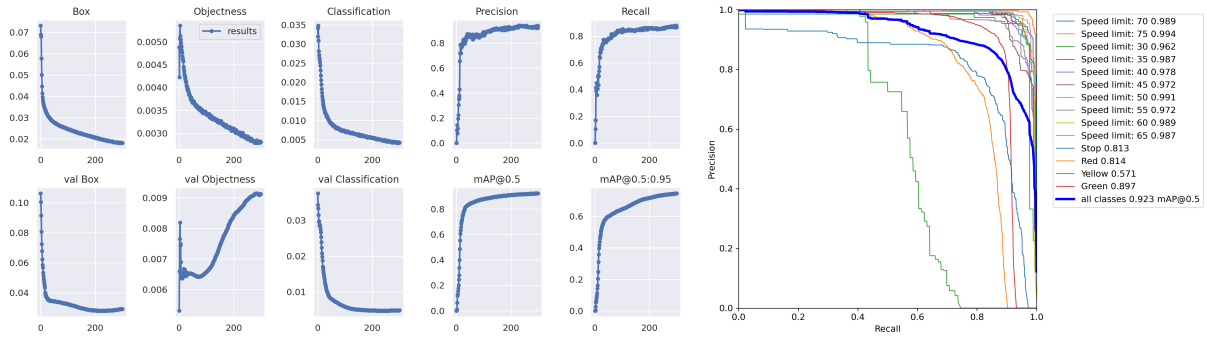


Figure 9.3: Graph of loss, and Precision-Recall for original model

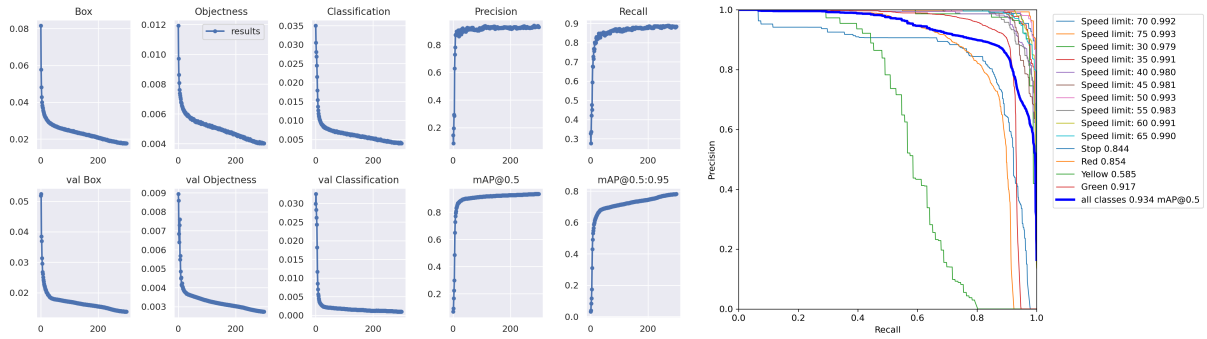


Figure 9.4: Graph of loss, and Precision-Recall for modified model

For both Figure 9.3 and 9.4, the first three columns of the left graph represent the loss for the box (regression), objectness (confidence), and classification. Among them, the first row represents the loss while training, and the second row represents the loss during validation.

In both Figure 9.3 and 9.4, the right graph indicates the precision-Recall curve. This is a method for evaluating performance in terms of the changing value of the threshold score (confidential

level). The threshold score means how much confidence the algorithm predicts and tells the target. This means that if the value is close to 1, this indicates that the algorithm has strong confidence in detecting the target.

Table 9.2 shows the detection results of the modified model. The table shows that the detecting speed limits are high which has over 90% for every score. This indicates that the detection rate of the algorithm is relatively high.

Table 9.2: Scores in terms of classes and total for Modified Model

classes	Precision(%)	Recall (%)	F1-Score (%)	mAP@0.5 (%)
Speedlimit: 30 mph	92.6	95.3	93.9	97.9
Speedlimit: 35 mph	96.9	94	95.4	99.1
Speedlimit: 40 mph	95	91.5	93.2	98
Speedlimit: 45 mph	96.7	92.1	94.3	98.1
Speedlimit: 50 mph	98.1	96.2	97.1	99.3
Speedlimit: 55 mph	89.5	93.1	91.3	98.3
Speedlimit: 60 mph	96.8	95.2	96	99.1
Speedlimit: 65 mph	94.5	95.4	95	99
Speedlimit: 70 mph	94.3	97.3	95.8	99.2
Speedlimit: 75 mph	98.3	97.6	98	99.3
Stop	80.5	84.2	82.3	84.4
Red	89.1	72.4	79.9	85.4
Yellow	89.1	49.1	63.3	58.5
Green	93.7	84.8	89	91.7
Total	92.8	88.4	90.6	93.4

The detection results for each class show that they have over 80% for mAP@0.5%. However,

the result for yellow light shows that the score of mAP@0.5% is 58.5% which is low. After the analysis of the dataset, we figured out that the lack number of data and the feature of yellow light in the image have an influence on the evaluation. In detail, first, compare to the other dataset of lights, there are a few number of images. Second, the shape of the yellow lights is different in the images. This means that some images contain the vertical shape of lights, while others contain the horizontal shape of lights. Moreover, the color of some traffic lights is yellow, while others are black. Therefore, when training, this may make the model confused.

9.2 Detection Result

9.2.1 Image Detection Result

The results for detecting targets are shown in Figure 9.5 and 9.6.

Figure 9.5: Detection result for Speed limit and Stop sign

Figure 9.6: Detection result for Traffic lights

In the results for speed limit in Figure 9.5, the algorithm detects the speed limit sign with high confidence. However, the results for a traffic light in Figure 9.6, show that the algorithm detects the traffic light with relatively lower confidence compared to speed limit signs. As far as we are concerned, we believe that the feature of traffic lights and the quality of the images are the reason that we got this result. First, a similar explanation for the result of the evaluation, the color and the shape of the traffic light are different so it might make a model to be confused. Second, most of the images of traffic lights are blurred and noisy. Thus, this model has strength for the detection of targets for blurred and noisy images, whereas this model has a weakness for the detection of targets for clear images. Still, it detects the traffic lights correctly.

9.2.2 Real-Time Detection Result

In order to prove whether the modified algorithm improved and works on real-time driving, both original and modified models are loaded in the Python code and run with the rosbag which has a recorded video. The results are shown in the figures below. The left image is the result of the original model and the right image is the result of the modified model.

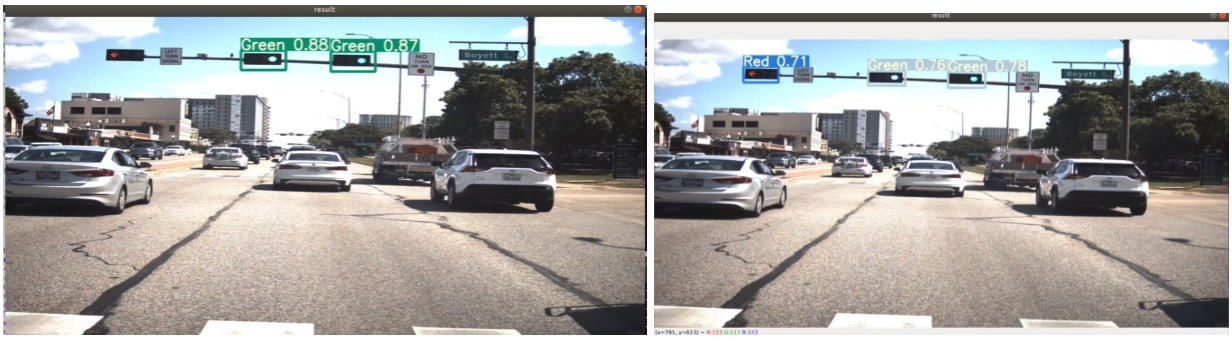


Figure 9.7: Start driving (0-10 mph) at section 1 in University Dr

In Figure 9.7, the original model only detects two green lights, but the modified model can detect both red light and green lights.

Figure 9.8: Driving at 40 mph at section 2 in University Dr

In Figure 9.8, the original model cannot detect any traffic signs, whereas the modified model does detect and classify the green light albeit the target is far from the vehicle.

Figure 9.9: Driving at 40 mph at section 3 in University Dr

In Figure 9.9, the original model has a misdetection and misclassification that it recognized the "Do Not Enter" sign as a stop sign. However, the modified model does not.

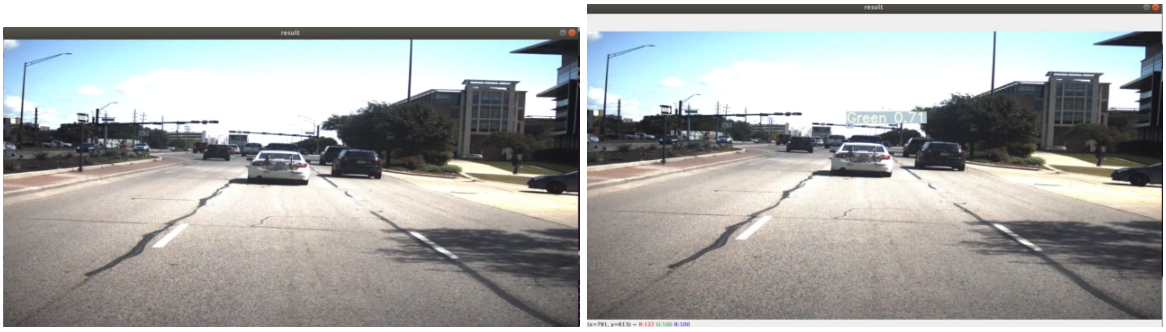


Figure 9.10: Driving at 40 mph at section 1 in University Dr

In Figure 9.10, the original model can't detect the traffic light, but the modified model can detect the green light barely.

Figure 9.11: Driving at 40 mph

In Figure 9.11, the original model detects nothing, but the modified model can detect the green light at a far distance.

Figure 9.12: Stop (0 mph)

In Figure 9.12, the original model can't recognize red lights. On the other hand, the modified model recognizes red lights.

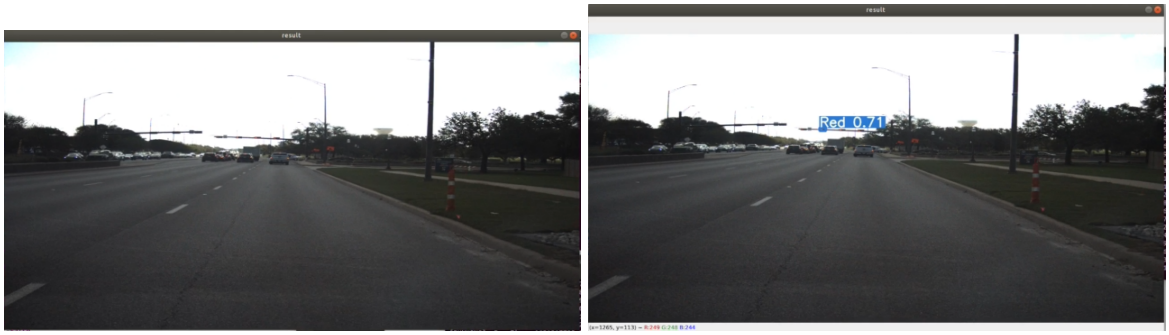


Figure 9.13: Driving at 40 mph

In Figure 9.13, the original model can't find any traffic light, but the modified model can find the red light which is at a long distance.

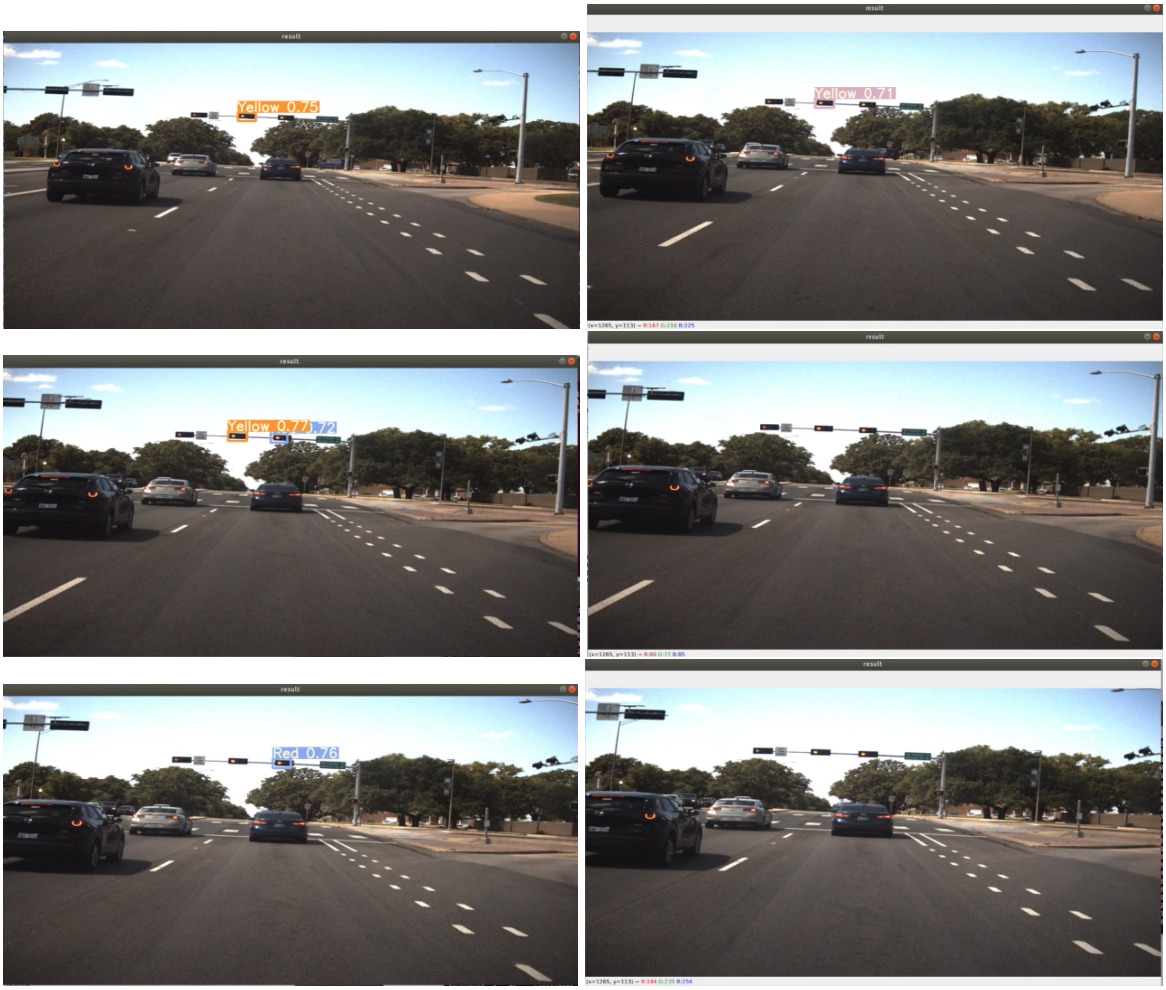


Figure 9.14: Slowing down (10-0 mph)

In Figure 9.14, both model has a problem with detecting red light when the sunlight is strong. However, as the distance between the vehicle and traffic lights is getting closer, the original model is more accurate than the modified model. In other words, the original model is strengthened than the modified model for the sunlight.

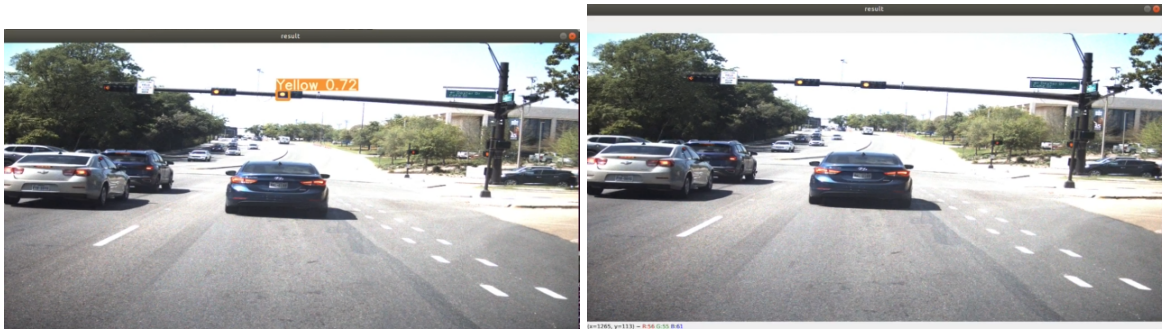


Figure 9.15: Slowing down (5-0 mph)

In Figure 9.15, similar to the previous result, both the original model and the modified model have difficulty of detecting red lights in strong sunlight. The original model interprets the red light as yellow light. Moreover, the modified model can't detect the red light.

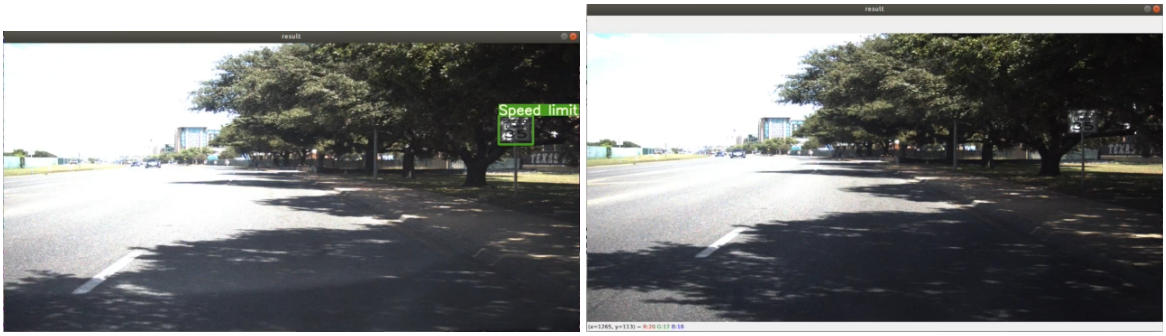


Figure 9.16: Driving at 35 mph

In Figure 9.16, it shows that the original model is better than the modified model when the light does not exist. The left image shows that the original model detects the sign barely. On the other hand, the right image shows that the performance of the modified model is weak when the sign is shaded.

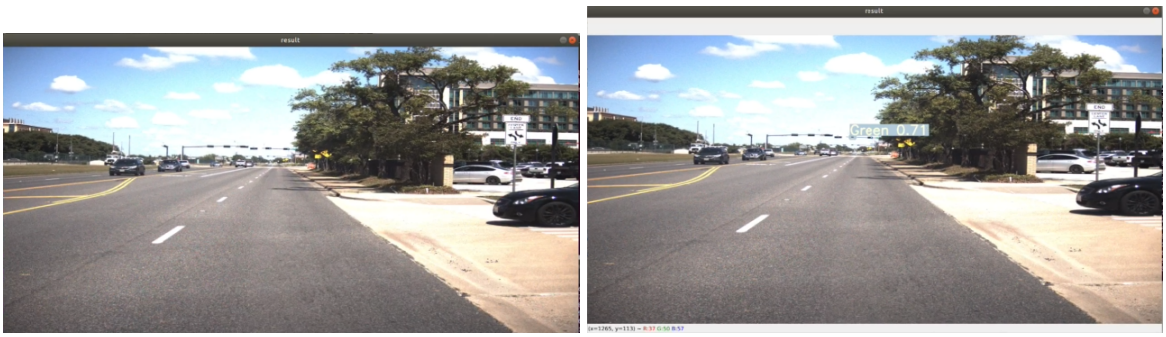


Figure 9.17: Driving at 35 mph

In Figure 9.17, the modified model has better performance than the original model for detecting the green light at a long distance.

Figure 9.18: Slowing down (20-5 mph)

In Figure 9.18, it shows that the modified model detects red lights regardless of the distance, whereas, the original model has a deflect that it could not detect red lights for both cases.

Figure 9.19: Start driving (0-5 mph)

In Figure 9.19, the original model misinterprets the traffic sign, but the modified model does not misdetect and misclassify the target.

Figure 9.20: Slowing down (5-0 mph)

In Figure 9.20, when the vehicle slows down and stops, the modified model detects and classifies the red light, but the original model can't detect any lights.

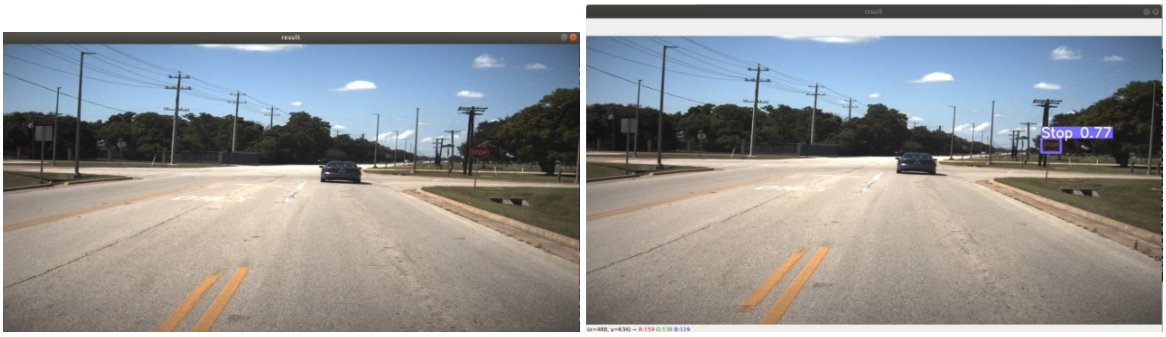


Figure 9.21: Slowing down (10-0 mph)

In Figure 9.21, the original model has a flaw that it could not detect the stop sign. However, the modified model detects and classifies the stop sign correctly.

Figure 9.22: Driving at 75 mph

In Figure 9.22, the upper images show that both models detect the yellow signs, but the difference is that the modified model detects all the yellow signs, but the original model detects only one

yellow light. And the bottom images show that the modified model detects all the yellow signs, whereas, the original model does not.

Figure 9.23: Driving at 75 mph

In Figure 9.23, the original model misdetect and misclassify the target, but the modified model does not make a wrong prediction.

To summarize the overall results, regardless of speed, we found out that the modified model detects traffic lights which the original model could not. Although the confidence in detecting targets is not high, we found out that the modified model is better at detecting traffic lights. Moreover, the modified model can detect the target which exists at a long distance. Even though the modified model barely detects traffic lights, it is helpful to prepare the action that the car suppose to take. Furthermore, the original model has misdetection and misclassification, whereas, the modified model does not. This is because, while the calculation of training images and testing rosbag, the activation function produces the negative value which will reduce the bias by pushing the mean close to 0. In addition, gradients will not be able to vanish and be transmitted to the following calculation.

However, there is a portion that didn't improve. In some cases, the modified model misdetect and misclassify the target depending on the brightness and darkness. When the sunlight is strong, the camera captures the red light as yellow light. So, we assume that the model gets confused to detect and classify the target when the sunlight is strong. Moreover, when the speed limit sign

is under the tree, which means that it is shaded, the modified model can't detect and classify the speed limit sign, whereas the original model does. After the analysis of the equation of the activation function, we figured out that the final output is saturated for large negative values. In other words, the output is exploded such that the output is out of range. As a result, the model cannot predict the target.

10. CONCLUSION

In this paper, we focused on studying and implementing the object detection algorithm for autonomous driving. To develop the model, a massive amount of images for traffic signs and lights that are currently used in the United States of America is required. And the goal is to improve the performance of the detection of traffic signs and lights in real-time driving.

After the comparison of the YOLO series, we decided to use YOLOv7 as the method for this project. In the experiment, we compare the original model to the modified model based on evaluating values, testing images, and testing videos by running rosbag. As a result of the evaluation, we found out that the mAP@0.5 of the modified model is improved by 0.9% compared to the original model. In addition, through a comparison of testing videos by running rosbag, we found out that the modified model detects and classifies targets fast and accurately albeit it is located at a long distance. However, we have noticed that there is a trade-off that the modified model has a weakness for brightness and darkness. Even though there is no critical difference between the original and modified model for comparing testing images, we conclude that the modified model is more useful than the original model in real-time driving.

Object detection and classification are important tasks for autonomous driving. As the algorithm runs in real-time, there are some factors that influence it such as brightness, camera resolution, etc. Under the circumstance that we are given, we conduct the project to improve the performance of the model. This study could be applied to other future work related to real-time autonomous systems.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[3] A. Rahman, Y. Lu, and H. Wang, "Performance evaluation of deep learning object detectors for weed detection for cotton," *Smart Agricultural Technology*, vol. 3, p. 100126, 2023.

[4] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[6] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, 2023.

[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–I, 2001.

[8] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, 05 2004.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[10] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[11] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2241–2248, 2010.

[12] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[13] F. Sultana, A. Sufian, and P. Dutta, "A review of object detection models based on convolutional neural network," *Intelligent computing: image processing based applications*, pp. 1–16, 2020.

[14] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2017.

[15] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

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

[17] G. Jocher, "Yolov5 by ultralytics."

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21–37, Springer, 2016.

[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017.

[20] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4203–4212, 2018.

[21] K. Grauman and T. Darrell, "The pyramid match kernel: discriminative classification with sets of image features," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1458–1465 Vol. 2, 2005.

[22] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2169–2178, 2006.

[23] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[24] G. Prabhakar, B. Kailath, S. Natarajan, and R. Kumar, "Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving," 10 2017.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[26] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge 2012 (voc2012)."

[27] S. M. Alhabshee and A. U. b. Shamsudin, "Deep learning traffic sign recognition in autonomous vehicle," in *2020 IEEE Student Conference on Research and Development (SCOReD)*, pp. 438–442, 2020.

[28] H.-K. Jung and G.-S. Choi, "Improved yolov5: Efficient object detection using drone images under various conditions," *Applied Sciences*, vol. 12, no. 14, 2022.

[29] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, "Designing network design strategies through gradient path analysis," *arXiv preprint arXiv:2211.04800*, 2022.

[30] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8759–8768, 2018.

APPENDIX A

ORIGINAL YOLOV7 APPENDIX

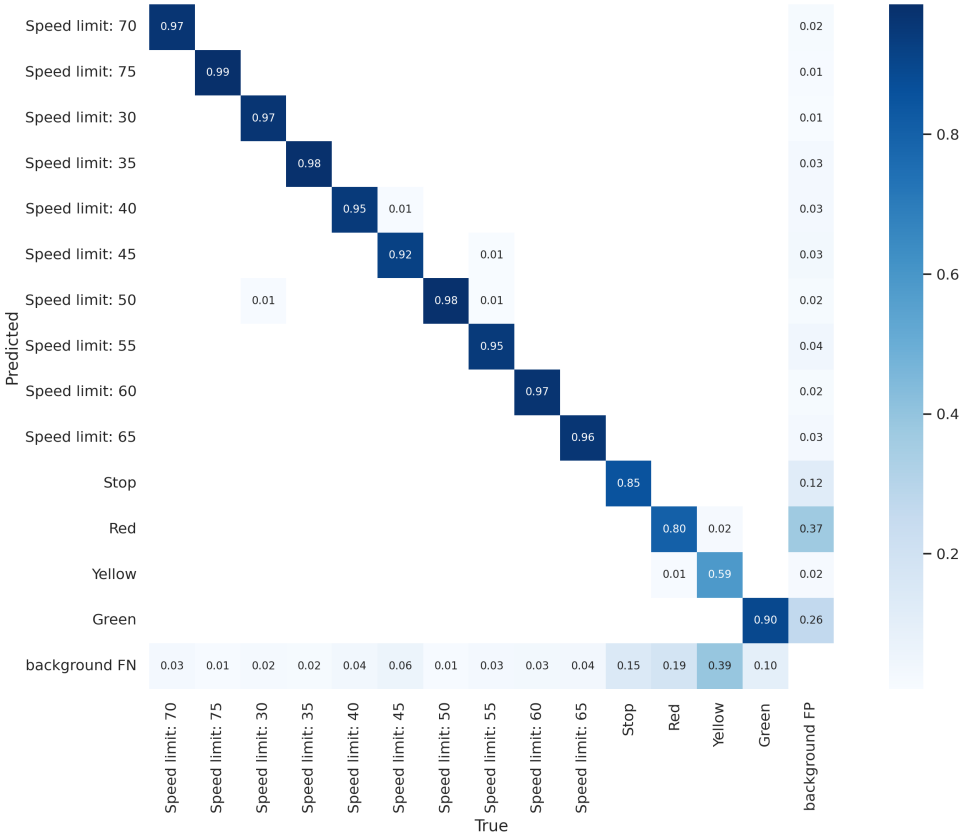


Figure A.1: Confusion matrix of Original YOLOv7

Figure A.2: ground truth for original YOLOv7

Figure A.3: predicted result for original YOLOv7

Figure A.4: training batch for original YOLOv7

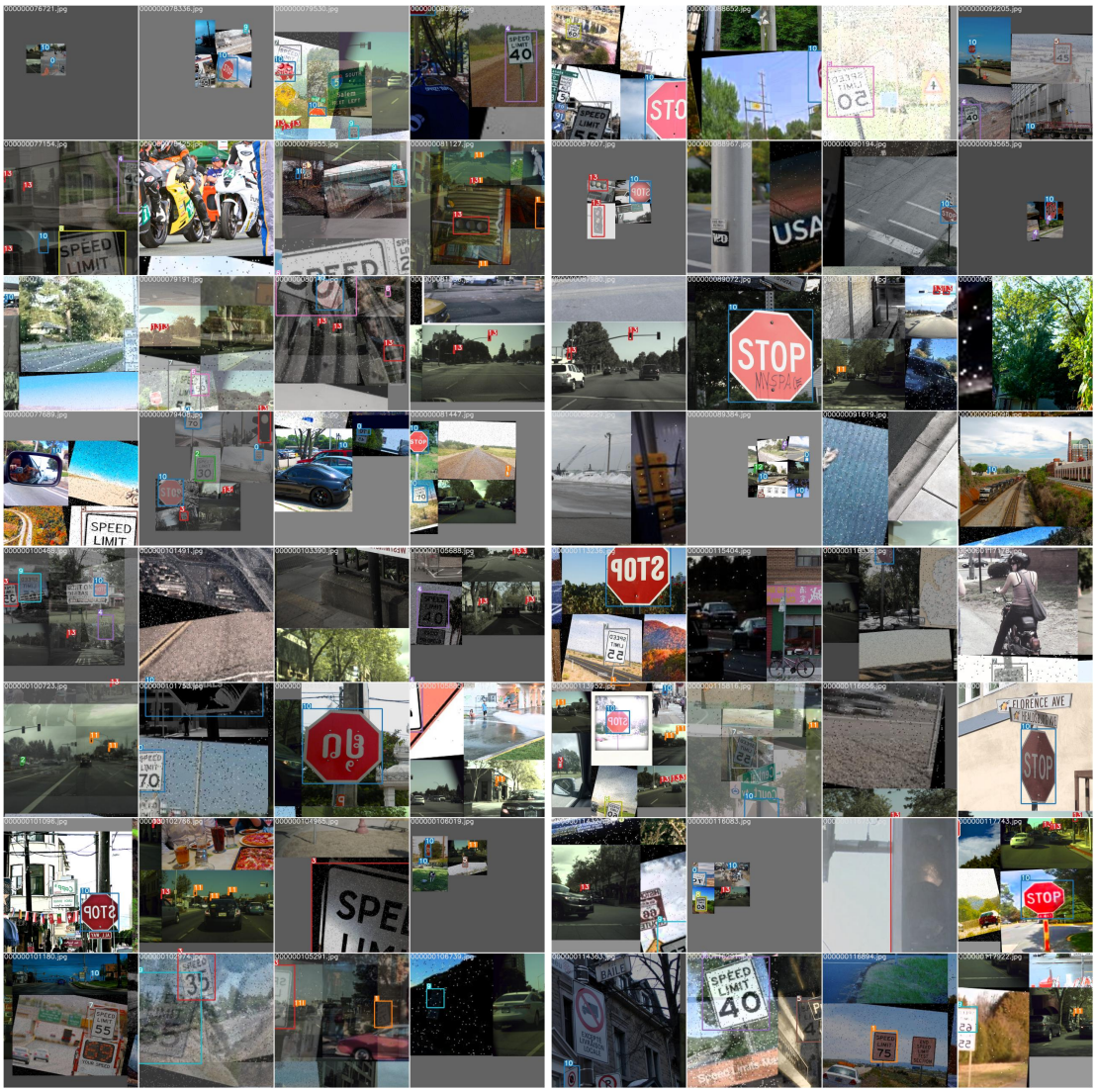
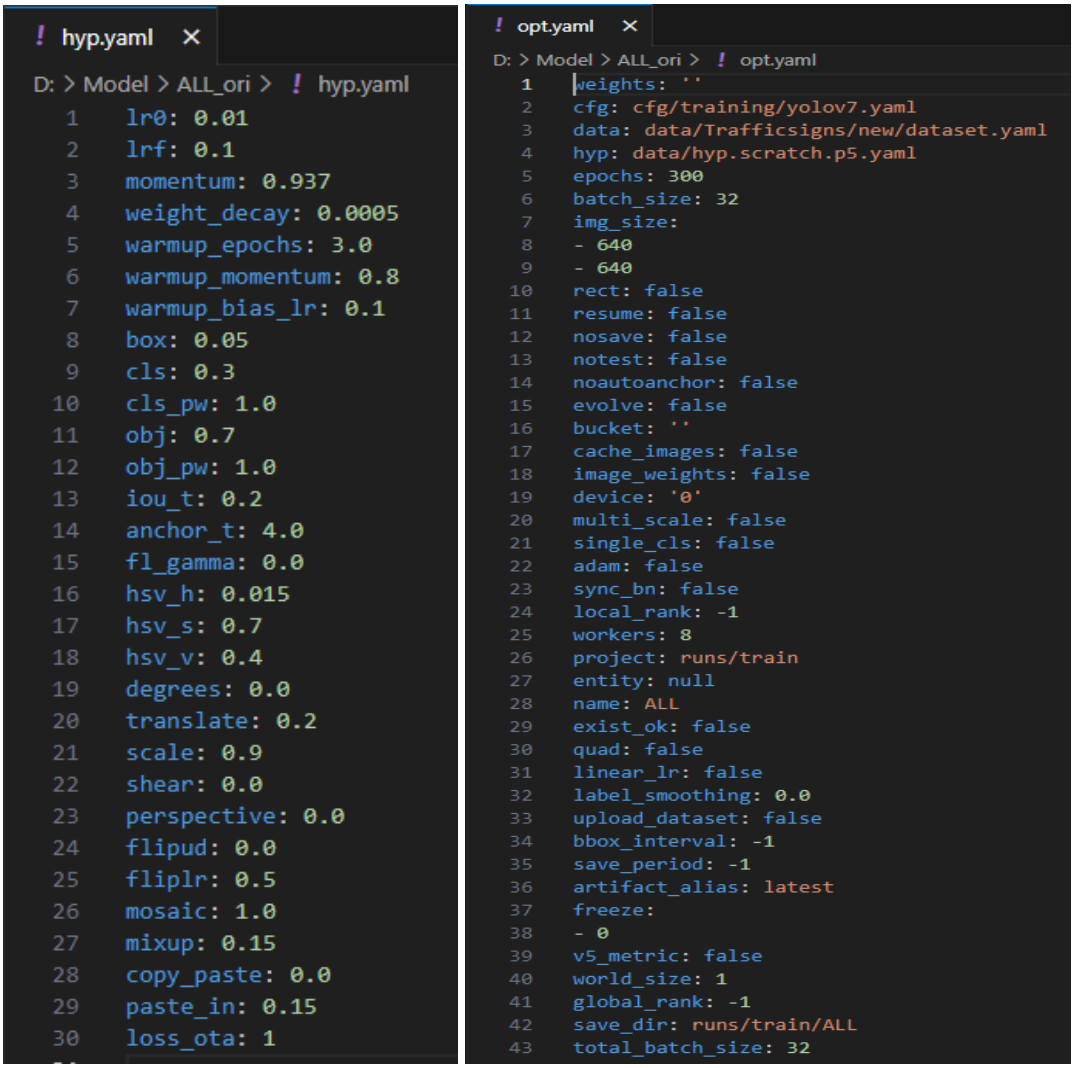


Figure A.5: training batch for original YOLOv7



hyp.yaml

D: > Model > ALL_ori > hyp.yaml

```yaml
lr0: 0.01
lrf: 0.1
momentum: 0.937
weight_decay: 0.0005
warmup_epochs: 3.0
warmup_momentum: 0.8
warmup_bias_lr: 0.1
box: 0.05
cls: 0.3
cls_pw: 1.0
obj: 0.7
obj_pw: 1.0
iou_t: 0.2
anchor_t: 4.0
fl_gamma: 0.0
hsv_h: 0.015
hsv_s: 0.7
hsv_v: 0.4
degrees: 0.0
translate: 0.2
scale: 0.9
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.5
mosaic: 1.0
mixup: 0.15
copy_paste: 0.0
paste_in: 0.15
loss_ota: 1
```

opt.yaml

D: > Model > ALL_ori > opt.yaml

```yaml
weights: ''
cfg: cfg/training/yolov7.yaml
data: data/Trafficsigns/new/dataset.yaml
hyp: data/hyp.scratch.p5.yaml
epochs: 300
batch_size: 32
img_size:
- 640
- 640
rect: false
resume: false
nosave: false
notest: false
noautoanchor: false
evolve: false
bucket: ''
cache_images: false
image_weights: false
device: '0'
multi_scale: false
single_cls: false
adam: false
sync_bn: false
local_rank: -1
workers: 8
project: runs/train
entity: null
name: ALL
exist_ok: false
quad: false
linear_lr: false
label_smoothing: 0.0
upload_dataset: false
bbox_interval: -1
save_period: -1
artifact_alias: latest
freeze:
- 0
v5_metric: false
world_size: 1
global_rank: -1
save_dir: runs/train/ALL
total_batch_size: 32
```

Figure A.6: Hyper-parameter(left) and the setting(right) for original YOLOv7

Table A.1: Scores in terms of classes and total for Original Model

classes	Precision(%)	Recall (%)	F1-Score (%)	mAP@0.5 (%)
Speedlimit: 30	95.3	93.7	94.5	96.2
Speedlimit: 35	95.3	94	94.7	98.7
Speedlimit: 40	95.8	91.5	93.6	97.8
Speedlimit: 45	96.3	90.3	93.2	97.2
Speedlimit: 50	97	95.4	96.2	99.1
Speedlimit: 55	89.5	93	91.2	97.2
Speedlimit: 60	96.4	95.3	95.9	99.1
Speedlimit: 65	96	93	94.5	99.72
Speedlimit: 70	95.4	95.5	95.5	98.9
Speedlimit: 75	98.5	97.6	98.1	99.4
Stop	78.3	79.9	79.1	81.3
Red	89.6	67.9	77.3	81.4
Yellow	95	43.4	59.6	57.1
Green	93.7	84.8	89	89.7
Total	93.7	86.6	90	92.3

APPENDIX B

MODIFIED YOLOV7 APPENDIX

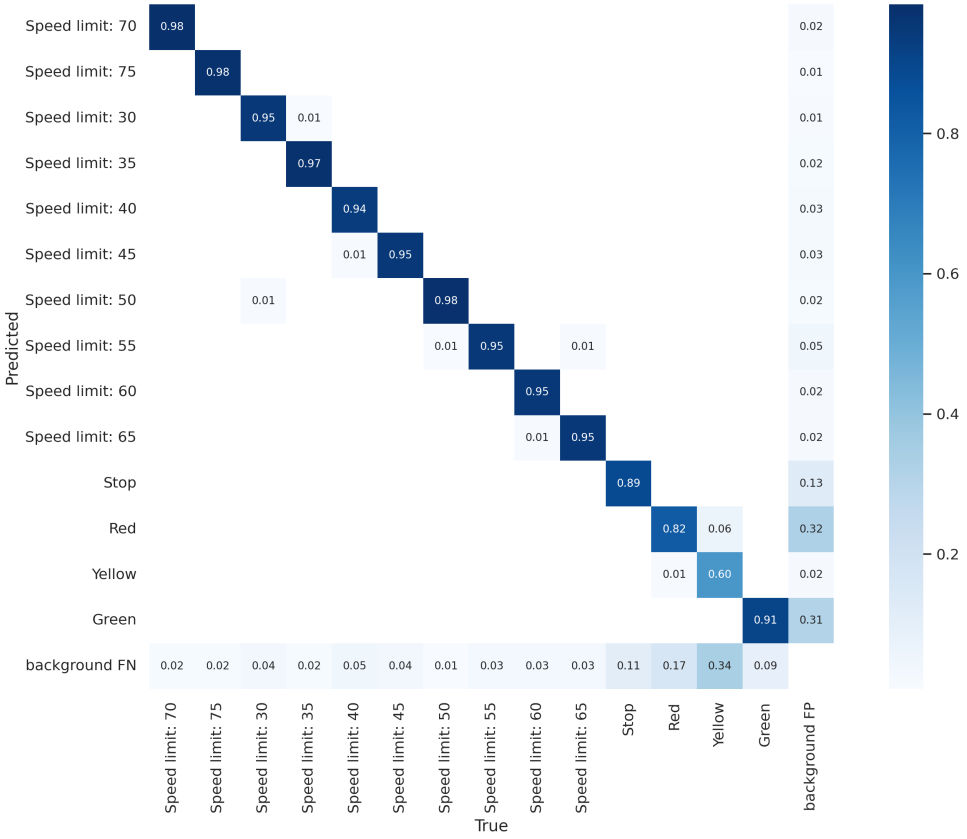


Figure B.1: Confusion matrix of Modified YOLOv7

Figure B.2: ground truth for Modified YOLOv7

Figure B.3: training batch for Modified YOLOv7

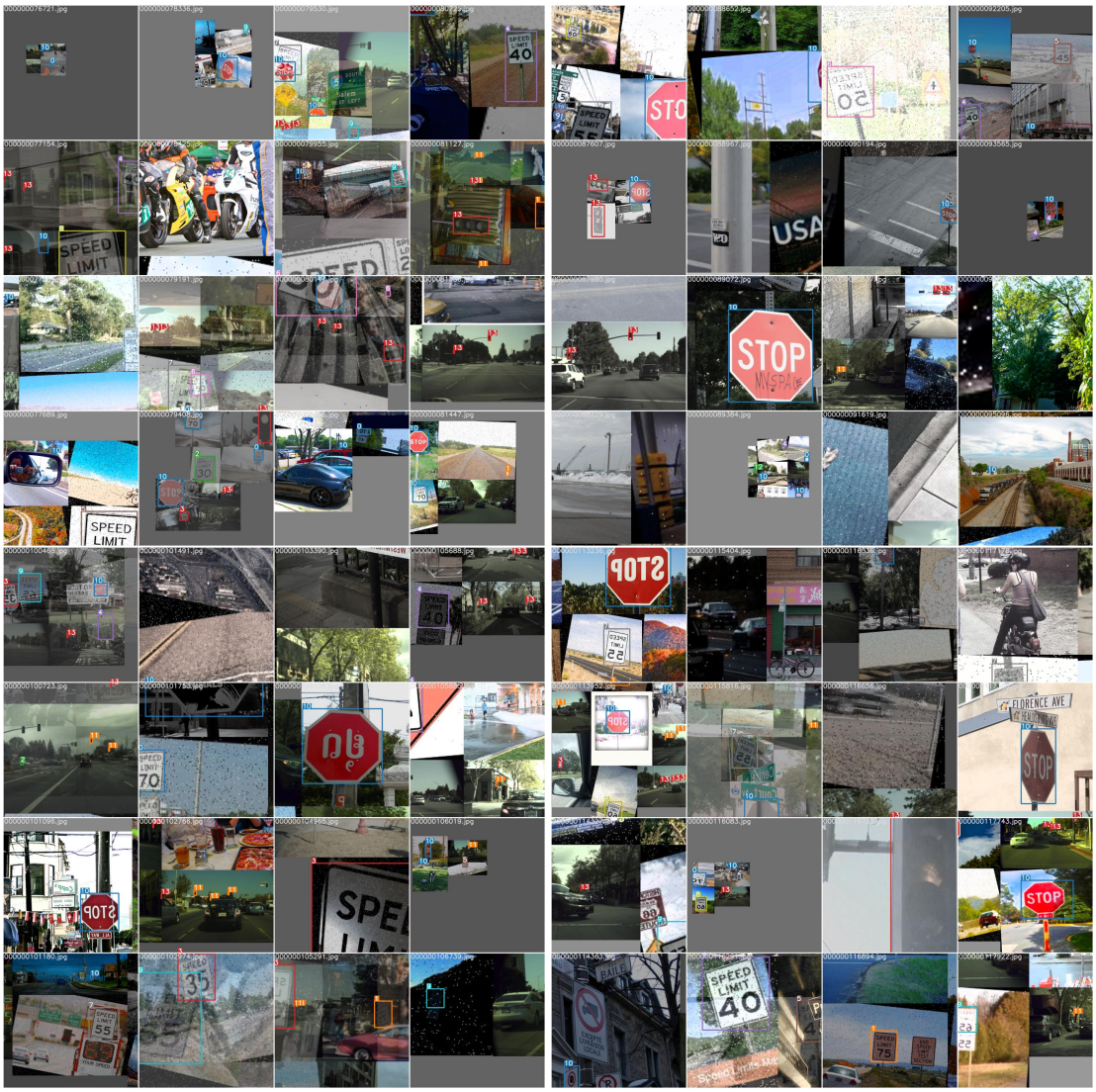
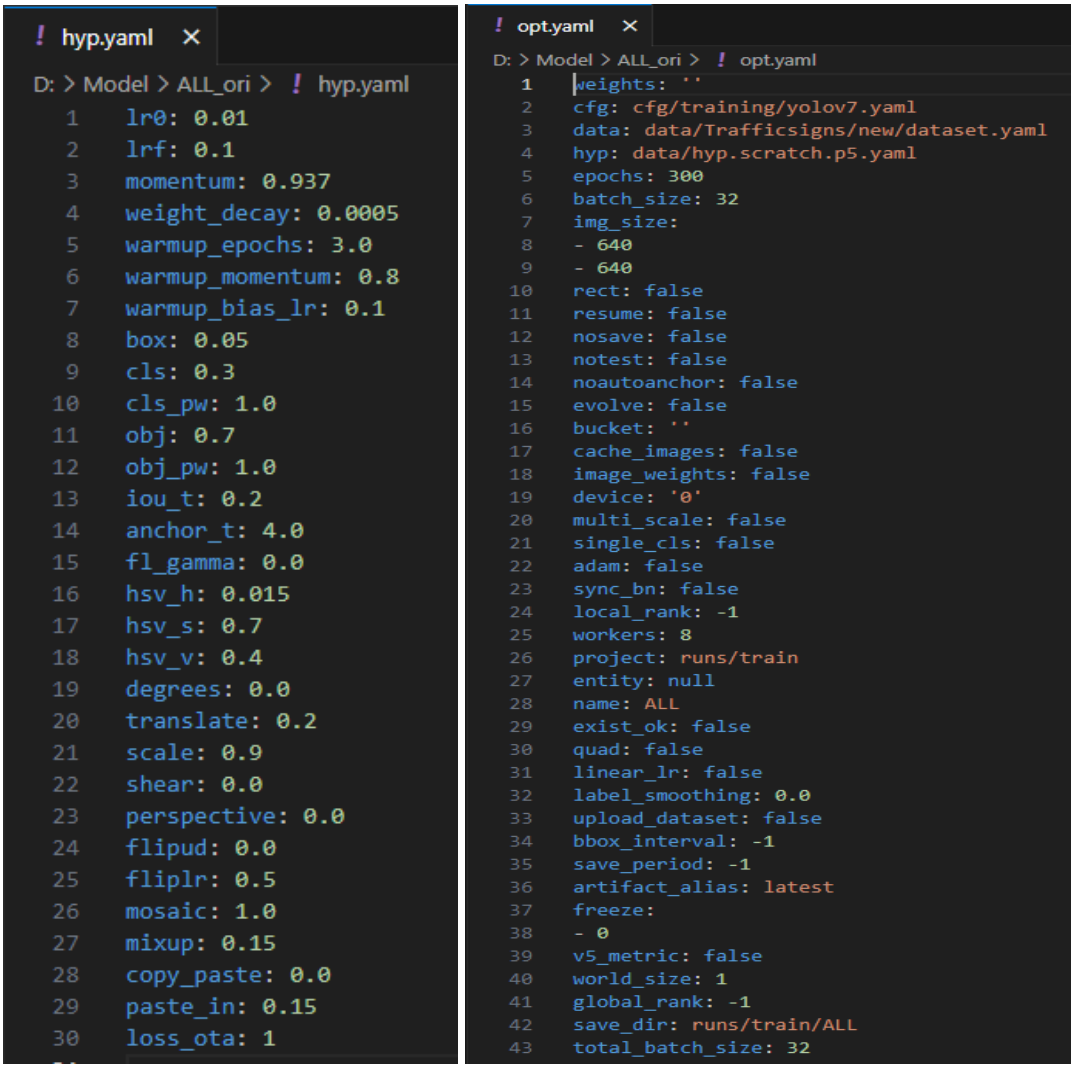


Figure B.4: training batch for Modified YOLOv7



hyp.yaml

D: > Model > ALL_ori > hyp.yaml

```yaml
lr0: 0.01
lrf: 0.1
momentum: 0.937
weight_decay: 0.0005
warmup_epochs: 3.0
warmup_momentum: 0.8
warmup_bias_lr: 0.1
box: 0.05
cls: 0.3
cls_pw: 1.0
obj: 0.7
obj_pw: 1.0
iou_t: 0.2
anchor_t: 4.0
fl_gamma: 0.0
hsv_h: 0.015
hsv_s: 0.7
hsv_v: 0.4
degrees: 0.0
translate: 0.2
scale: 0.9
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.5
mosaic: 1.0
mixup: 0.15
copy_paste: 0.0
paste_in: 0.15
loss_ota: 1
```

opt.yaml

D: > Model > ALL_ori > opt.yaml

```yaml
weights: ''
cfg: cfg/training/yolov7.yaml
data: data/Trafficsigns/new/dataset.yaml
hyp: data/hyp.scratch.p5.yaml
epochs: 300
batch_size: 32
img_size:
- 640
- 640
rect: false
resume: false
nosave: false
notest: false
noautoanchor: false
evolve: false
bucket: ''
cache_images: false
image_weights: false
device: '0'
multi_scale: false
single_cls: false
adam: false
sync_bn: false
local_rank: -1
workers: 8
project: runs/train
entity: null
name: ALL
exist_ok: false
quad: false
linear_lr: false
label_smoothing: 0.0
upload_dataset: false
bbox_interval: -1
save_period: -1
artifact_alias: latest
freeze:
- 0
v5_metric: false
world_size: 1
global_rank: -1
save_dir: runs/train/ALL
total_batch_size: 32
```

Figure B.5: Hyper-parameter(left) and the setting(right) for Modified YOLOv7

APPENDIX C

STRUCTURE APPEDNIX

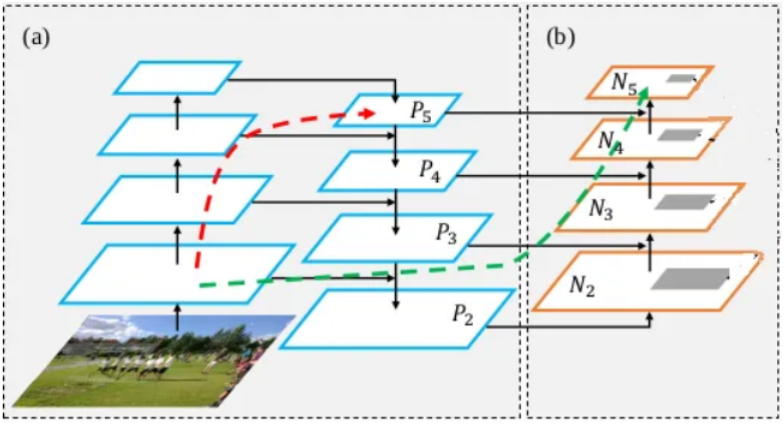


Figure C.1: Structure of PAFPN