Object Detection and Tacking for Autonomous Driving

Oguzcan KARADENIZ

Final Dissertation for Master of Science

from the

University of Surrey

Department of Electronic Engineering

Faculty of Engineering and Physical Sciences

University of Surrey

Guildford, Surrey, GU2 7XH, UK

September 2022

Supervised by: Dr Saber Fallah

DECLARATION OF ORIGINALITY

I confirm that the project dissertation I am submitting is entirely my own work and that any material used from other sources has been clearly identified and properly acknowledged and referenced. In submitting this final version of my report to the JISC anti-plagiarism software resource, I confirm that my work does not contravene the university regulations on plagiarism as described in the Student Handbook. In so doing I also acknowledge that I may be held to account for any particular instances of uncited work detected by the JISC anti-plagiarism software, or as may be found by the project examiner or project organiser. I also understand that if an allegation of plagiarism is upheld via an Academic Misconduct Hearing, then I may forfeit any credit for this module or a more severe penalty may be agreed.

Object Detection And Tacking For Autonomous Driving

Oguzcan KARADENIZ

Author Signature:  Date: 06/09/2022

Supervisor’s name: Dr Saber Fallah

WORD COUNT

Number of Pages: 66

Number of Words: 12722

ABSTRACT

The perception part, which is very necessary for autonomous vehicles, has been examined. The details of a particular architecture were explored to use Object detection, classification and tracking methods. In addition, research was carried out on the data set required for the training process. In this project, we will focus on two YOLO algorithms, which are Scalled-YOLOv4 and YOLOv7. Moreover, we will train them with two different datasets, which are KITTI and WAYMO datasets. After training, we will try to improve the accuaracy by using fine tunining and freezing layers technics. In addition some significant graphs that reflet our training processes results will be examined. Finally, we will simulate the best trained model weight.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

It is an unavoidable reality that technological advancements contribute significantly to the development of autonomous cars. A lot of work has been done in this field and continues to be done. The 5 steps required for vehicles to move autonomously are as follows. They are, in order, Perception, Localization and Mapping, Path Planning, Decision Making, and finally Vehicle Control. Autonomous vehicles analyse environmental factors and store them for detecting, classifying and tracking these detected objects and then using this data in other stages. This stage is the Perception stage, the first step, and allows autonomous vehicles to explore their environment and to act completely autonomously by feeding other steps with this discovery data. In short, it is an extremely important step, and in this section, CNN, which is a subclass of neural networks, deep learning, which is a machine learning method, and computer vision techniques are used. In this report, Scaled-YOLO-v4 [1] and YOLO-v7-will be examined in detail and detailed information will be given on the detection, classification and tracking of objects.

Moreover, to train these algorithms, there is no doubt that we have a dataset that is specific for autonomous cars, which are, for example in this report, KITTI and WAYMO datasets. Then after some experiments and arranging the datasets for YOLO style, we will train these algorithms with some methods. After having some .pt files(trained weights), we will use them for simulation to evaluate them. Also, after some experiments, we tried some methods to get higher accuracy such as fine tuning and freezing some layers. As a summary, this is a report of both experimental and programming.

1.1 Background and Context

The primary objective of this project is to construct a neural network capable of performing object detection, classification, and tracking in the perception component, which is critical for autonomous driving, as well as to train and analyse the network using relevant data sets. These operations will be implemented some videos to simulate.

1.2 Scope and Objectives

1. Using neural network algorithms that is appropriate for the project's objectives and using it for object identification, categorization, and tracking.
2. With the agreed-upon data set, train the constructed neural network models with different methods.

3. To evaluate the performance, interpreting the graphs and data that are results of training processes and then depending on these types of information, try to get higher accuracy by trying some techniques.

4. After training process, compare these experiments and present the performance of trainings by simulating them.

1.3 Achievements

After the realization of this project, I think that I will gain a lot of knowledge and skills in the field of perception, which is very important for autonomous vehicles. Using CNN architecture on image processing in the field of Deep Learning, which is one of the most popular topics in today's world, will give me a lot of experience and knowledge. Therefore, I will have the opportunity to work in many existing or newly established companies for autonomous vehicles, which are the cars of the future.

1.4 Overview of Dissertation

This report will focus on the most important part of autonomous vehicles, the perception part. This part is divided into 3 parts, namely object detection, classification and tracking, and this project will focus on these parts. In order to realize these parts, the algorithm, dataset and simulation we need will be examined in detail and I will give my comments on what improvements will be made. Respectively, YOLO Algorithm and its specific features with some significant metrics that help us to understand the performance of training processes, datasets we used, Scalled-YOLOv4 and YOLOv7 and finally experimental results can be found in this report chapters.

2 BACKGROUND THEORY AND LITERATURE REVIEW

This section will define the YOLO algorithm., the details of techniques to get better accuracy, what Object Detection, Classification and Tracking mean, and finally the research on the dataset that is considered to be used.

2.1 YOLO Algorithm

Deep learning techniques for classification and regression are being used in two different ways. The first is the R-CNN, Fast R-CNN, and Faster R-CNN architectures, all of which use a two-stage algorithm. Frequently, this algorithm is divided into two pieces.[4] To generate Region Proposal, execute a selective search or use the Region Proposal Net (RPN), followed by classification and regression on Region Proposal. This technique is very accurate, but has a poor detection time. Another approach is the one-stage algorithm, which is referred to as SSD, YOLO, and other abbreviations.

This is a regression-based object detection approach that predicts the object border box and category likelihood score directly from the image using a single network. The detection time is enhanced since this technique does not employ RPN.[5] The identification rate for little targets, on the other hand, is not as excellent as with the two-stage method. The model's detection accuracy and speed have a direct impact on identification capability.

Some of the two-stage algorithms provides a bit more accurate results compared to one-stage algorithms. However, these are unfortunately having less speed in comparison of FPS. For this reason, we need to focus on one-stage algorithms. We have already mentioned about some algorithms that are YOLO and SSD. SSD can perform extremely quick, but this is inaccurate. As a result, two-stage algorithms are slow and from one-stage algorithms, SSD does not give correct results. When we looked at YOLO, it has the ideal balance of speed and accuracy. Therefore, we will focus on this algorithm and its derivatives.

2.2 Object Detection, Classification and Tracking

Object detection examines in two categories named by non-neural and neural network approaches. As an example of non-neural methods, SIFT,HOG and etc. can be given.

For neural network methods, R-CNN and its derivatives, SSD and YOLO and its variable versions that are the topic of this report can be given as examples.

Object detection is a technique used in computer vision for identifying objects in images and

videos. Object identification algorithms often make use of machine learning or deep learning to provide appropriate findings. When we look at images or videos, we can discover and pinpoint objects of interest in a few of seconds. The objective of object detection is to emulate this intelligence via the use of a computer.

To detect objects, various methods can be used and an example of these, the deep learning approach can be said. The popular and being able to give more accurate results architecture can be said as CNN (convolutional neural networks. And the most popular algorithms using CNN are YOLO and its derivatives (Single-Stage Networks), R-CNN and its derivatives (Two-Stage Networks) and etc. for object detection.

As we mentioned, single-stage detectors perform better in inference, but two-stage detectors obtain high accuracy in localization and object recognition. However, more accuracy does not always imply that this is the optimal course of action for achieving your objective. For instance, in real-world situations, we want high-speed detection results, such as autonomous driving.

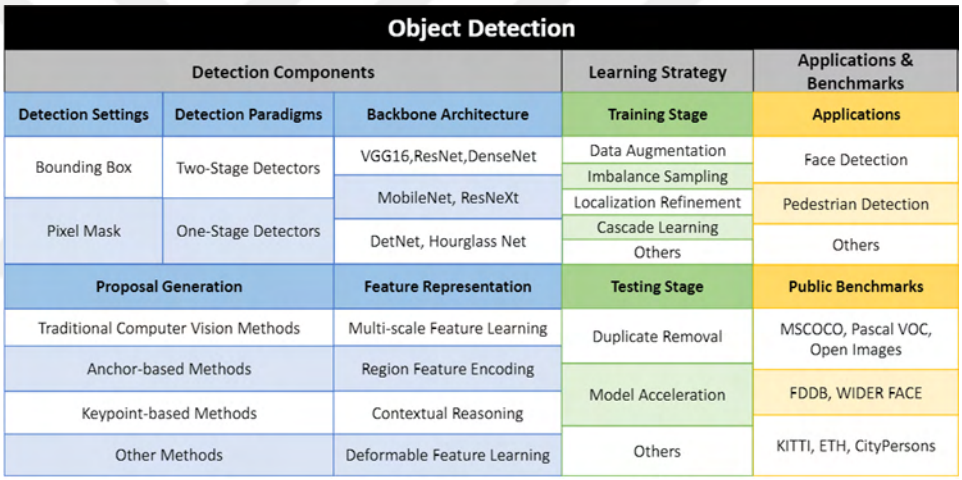


Figure 1

Multi object detection will be using in this project because we have various objects that we need to detect for every image from continuous real time video streaming.

In addition, object recognition includes some steps that are video sequence, object localization, object classification and as a result of this process, object detection. When we have one object needed to identify what this is we classify and then locate. And this process is called localization + classification. However, if we have one more object in a frame, the process name is object detection and this is called multi-object detection as well.

Moreover, another object detection area that will be determined according to the progress of the project is 3D object detection. Extending prediction to three dimensions allows the capture of an object's size, position, and orientation in its environment, which has applications in robotics and autonomous driving. While 2D object identification is well-known and commonly utilized in busi-

ness, 3D item detection from 2D images is a difficult challenge to solve owing to a lack of data and the variety of appearances and forms of objects within a category.

For self-driving car, most challenging part of detecting objects is 3D object detection. For this reason, object detection in autonomous cars is used to comprehend objects' every move, behavior, and intention, rather than merely detect them. As a result, while 2D object detection is important for autonomous cars, we get the most value from 3D object detection. However, 2D object detection is a part of 3D object detection so that having some capabilities first about 2D object detection is absolutely significant.

The task of classifying the objects detected in the frame according to their type of interest is defined as object classification. It's simply a matter of establishing the nature of the object. Identification of objects can be accomplished through the use of a variety of factors, including shape, motion, color, and texture. Depending on the parameter used, we may classify objects using shape, motion, color, or texture. We may have various objects that we need to classify them such as cars and traffic signs. For this reason, we have to use multi-label classifier to classify them separately.

Another thing that we need to talk about is object tracking. Object tracking part comes after the object detection or localization and classification of single object and so we can say that there are two method, which are single object tracking and multiple object detection. Single object tracking is perhaps the most well-known and straightforward of the tracking sub-problems. The goal is to simply lock on to a single object in the image and follow it until it leaves the frame. This form of tracking is simpler since the more difficult challenge of differentiating this object from others isn't there. In multiple object tracking, it is supposed to lock onto every object in the frame, identify particular every one of them, and track them until they exit the frame. In addition, if we want to track same class objects, we need to assign unique ID number to them. For example, we want, separately, to track 3 car that are in same class, we should tag them with different unique ID.

2.3 Datasets

In this project, 2 most famous datasets was used, which are KITTI and WAYMO. We can find various type of datasets for these datasets such as for 3D object detection. However, we will benefit from 2D object detection datasets of them.

1. WAYMO Dataset

   The Waymo Open Dataset is made up of high-resolution sensor data (camera and lidar) gathered by self-driving cars driven by Waymo Drivers under a range of circumstances. The research community can develop machine perception and self-driving technologies with the

help of this dataset, which is made available to the public. Waymo dataset divided into two categories which are motion and perception datasets. In this project, we will benefit from perception dataset. Waymo is little bit different dataset compared to other types of dataset. This dataset file type is ".tf". Actually, this is a video sequence with some types of data which are sensor data(from lidar and cameras) and labels(3D and 2D bounding boxes, key points, 2D-to-3D correspondence, 3D semantic segmentation and 2D video panoptic segmentation.

Total dataset size approximately 2TB but these files are splitted by lots of part so we can use one of them not to train 2TB dataset. In total of dataset includes 2,030 segments of 20s each, collected at 10Hz (390,000 frames) in diverse geographies and conditions. These types of data, such as sensor data, collected by 1 mid-range lidar, 4 short-range lidars, 5 cameras (front and sides). Also, for labels, we can say that there are 4 type of object classes which are vehicles, pedestrians, cyclists and signs. The bounding box labels are high-quality labels for camera data in 1,000 segments and also totally 11.8M bounding box labels with tracking IDs on camera data can be defined.

Figure 2

In the case of Waymo dataset, we can take a deep look at how labels are created because it is important part to have an accurate dataset. For example, we can talk for 3 types of classes out of 4 classes;For vehichles,

- There are labels on any item that can be identified as a vehicle and is at least partially visible.
- Trams and trains not labelled because they are not evaluated as a vehicle.
- Motorcycles is a type of vehicle so they are labelled as a vehicle.

For pedestrians,

- There are labels on any item that can be identified as a pedestrian and is at least partially visible.
- If people are walking or riding a scooters or ridind stakeboards, etc., they

are labelled as pedestrians.

- Pedestrians sitting inside a car are not labelled.
- If a person rides a biycles, it is labelled as a cyclist instead of a pedestrian.
- For example, people found in billboards, reflections are not labelled as a pedestrian.

For Cyclists;

- There are labels on any item that can be identified as a cyclists and is at least partially visible.
- If there is a biycle without a rider, it is not labelled.
- When a person start to get off the bicycle, this scenario is labelled as a pedestrian instead of cyclist.

Because this project depends on 2D object detection, classification and tracking, we will benefit from the camera datas and 2D bounding boxes labels. Therefore, we need to extract these datas from the whole file, ".tf". In this project, we extracted all cameras data and labels but we will use only the middle camera data. However, if we want to mix up them, we can and we can have width spectrum dataset but it will increase the training cost. Moreover, to have a target-base dataset by extracting dataset, there are several transform algorithms but they needed to have some modification on themselves. First, images are extracted from video sequence and their corresponding label data are extracting. However, these transformers only transform the types of data from Waymo to KITTI format. Therefore, after having types of data on KITTI format, we need to use other transform algorithms by modify them to convert COCO format.

When we looked at the split format of the dataset, this is 80:10:10(training, validation, test) split dataset and we can see their distribution in the case of 4 classes.

Table 1

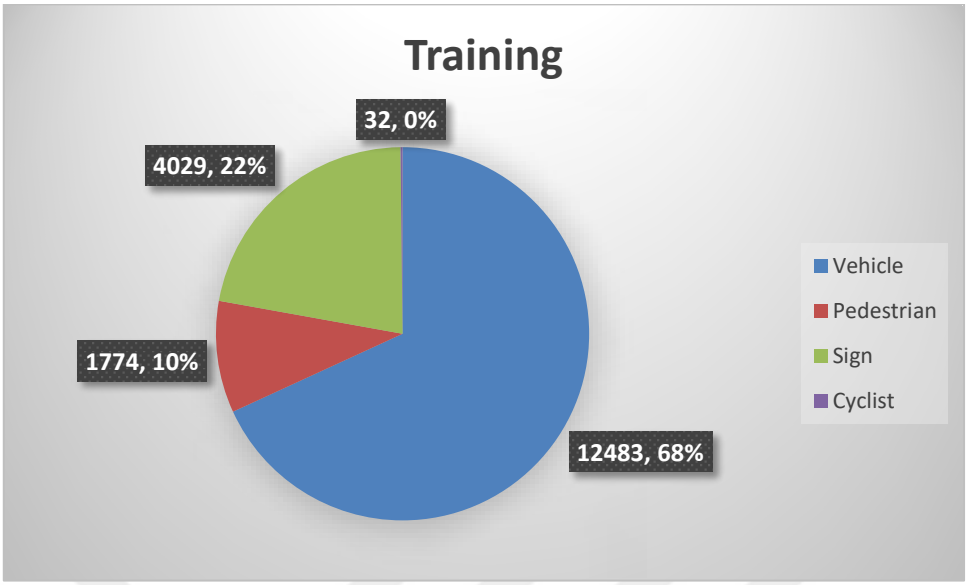


Table 2

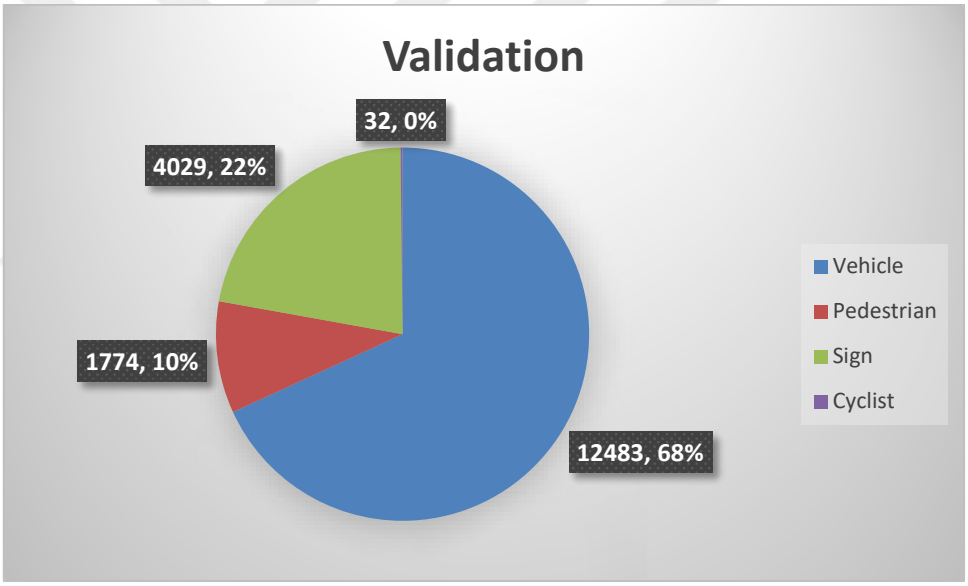
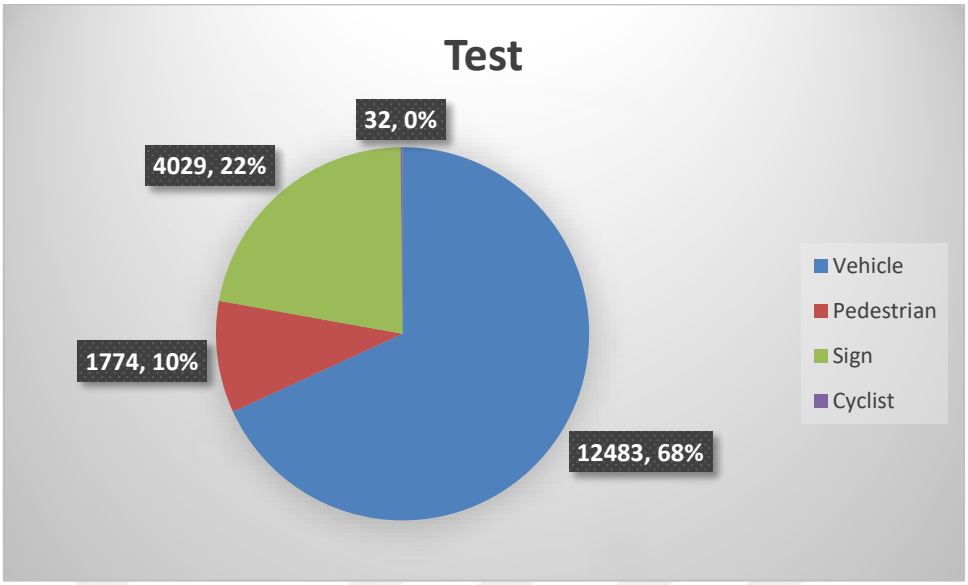


Table 3



2. KITTI Dataset

KITTI is a frequently used dataset in the field of mobile robotics and self-driving vehicles.[7] It is constructed from hours of traffic conditions taken with a variety of sensor modalities, including high-resolution RGB and grayscale stereo cameras. This dataset contains 7481 training images, 7518 test images, and, in its standard form, 80.256 labelled objects[7]. These images have color and stored as png format. Despite its extensive usage, the dataset is deficient in terms of ground truth for semantic segmentation. The advantage of this dataset is that various images have been taken in various places including city, residental, road, campus and etc. and we can see from the image below.

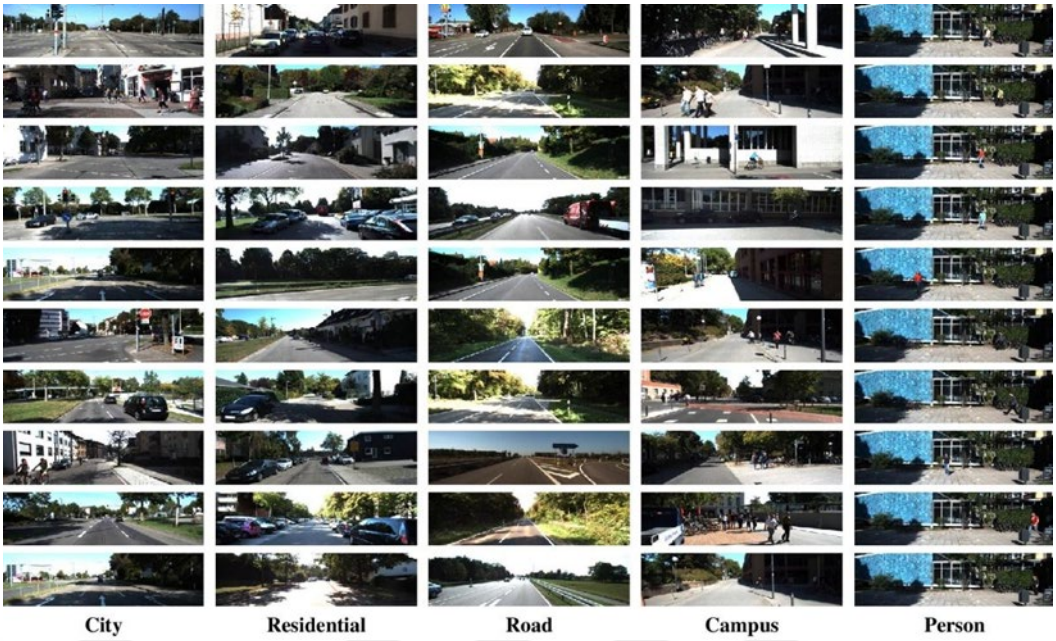


Figure 3

With KITTI dataset, we can train 8 classes of objects, which are respectively car, van, truck, pedestrian, person_sitting, cyclist, tram and Misc. In this project, to have 3 type of dataset, which are training, validation and test, we split the original version of KITTI dataset from 2 type, training and testing, to 3 type that we mentioned. As a splitting ratio, 80:10:10(training, validation and test) structure was used. Also, to analyse the dataset split and analyse the classes in these splits, we can benefit the graph.

Table 4

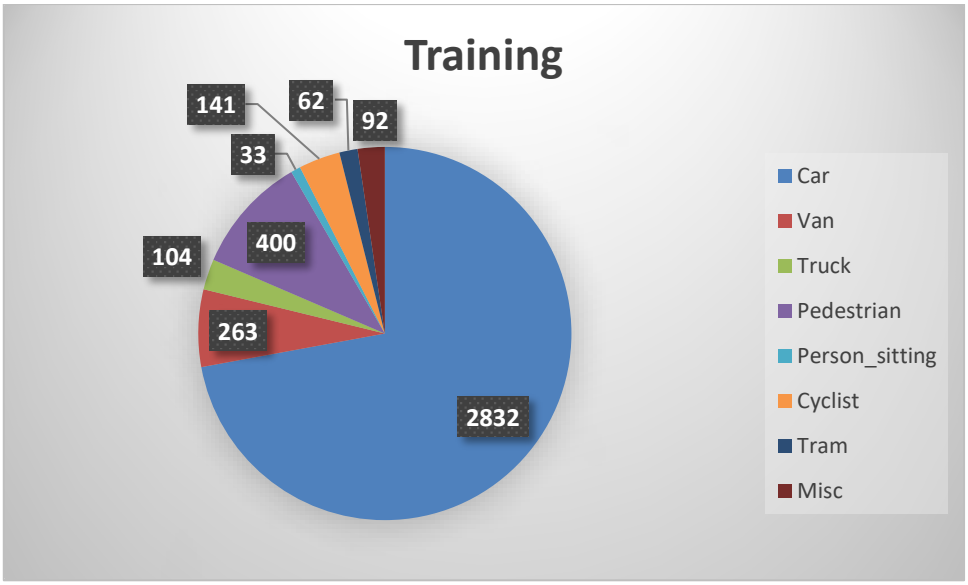


Table 5

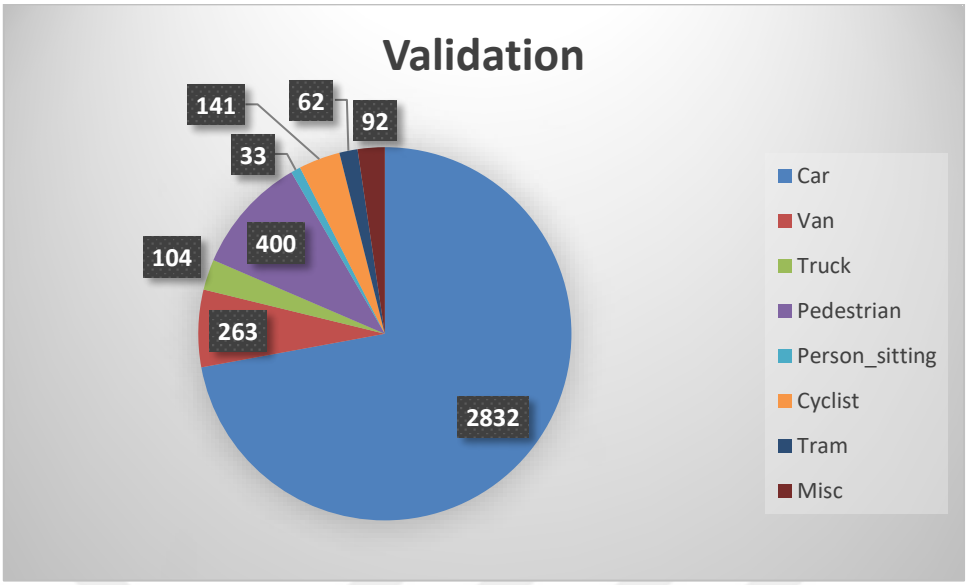
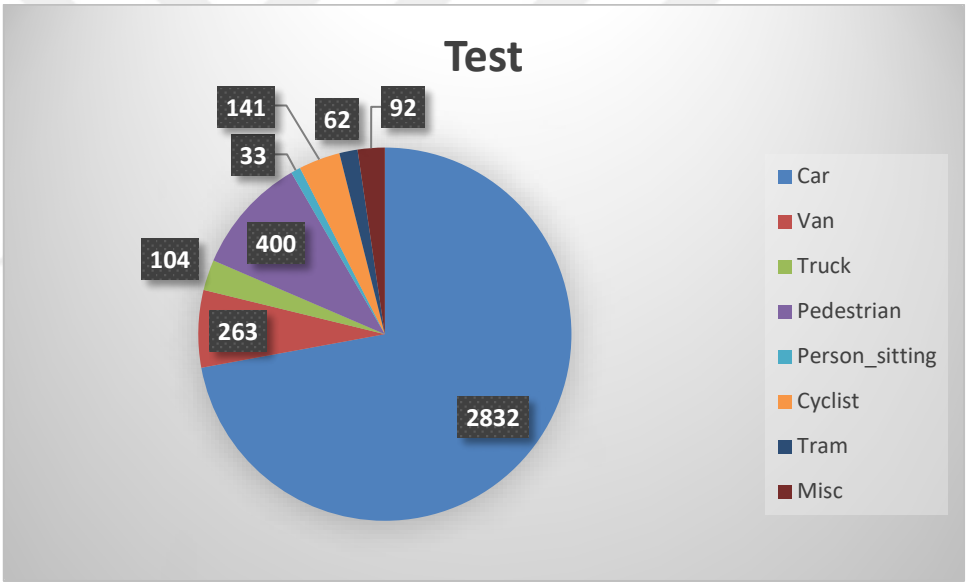


Table 6



Moreover, because we are using YOLO algorithms, we needed to arrange the KITTI label format to COCO format. Because, YOLO algorithms are suitable to use with COCO dataset format. To change KITTI label format to COCO label format, we benefit from some transformer algorithms but by modifying them because they don't work correctly. As an example of KITTI format and COCO format, we can take a look at these structure below.

Firstly, this is a KITTI label format;

Pedestrian 0.00 **0** -0.20 **712.40** 143.00 **810.73** 307.92 **1.89** 0.48 **1.20** 1.84 **1.47** 8.41 **0.01**

These values are corresponding to various things;

Table 7

Type	Truncated	Occluded	Alpha	Bbox	Dimensions	Location	Rotation	score
1	1	1	1	4	3	3	1	1

When we looked at the COCO label format;

2 0.7660390625000001 **0.5552890625** 0.006276041666666643 **0.018843750000000003**

Table 8

type	Bbox_x	Bbox_y	Bbox_width	Bbox_height
1	1	1	1	1

We can see that our label format will be decreased from 15 values 5 values and with this label format YOLO is ready to training.

2.4 Summary

In this chapter, we first looked at YOLO algorithm that is one-stage algorithm and we elaborate why we are using it instead of two-stage object detectors. Secondly, we explained what the object detection, classification and tracking terms is. Finally, we took a deep look at the datasets that we will use to train YOLO algorithms.

3 IN-DEPTH TECHNICAL SURVEY OF THE YOLO ALGORITHM

YOLO is a real-time object identification system based on convolutional neural networks (CNN), and it is capable of detecting, tracking, and classifying objects.. This technique gives high accuracy while detecting object faster. This method may be found in a variety of subjects, including autonomous driving for traffic signs, cars, pedestrian identification, and so on.

Three categories may be studied while examining YOLO: bounding box regression, residual blocks, and intersection over union (IOU).

These days, the most efficient and balanced algorithm for both accuracy and speed is Scaled-YOLOv4 and YOLOv7. Not only these things, but also these algorithms comes with new approaches to obtain more accurate on detecting small objects. There is no doubt that small objects in traffic conditions can be observed such as traffic lights. So, we will use this algorithms. And the following part, we will look at these algorithms with more detail.

Table 9

	YOLOv3	YOLOv4	Scaled YOLO v4	YOLOv7
NN Architecture	Fully convolution	Fully convolution	Fully convolution	Fully convolution
Neck	FPN	SPP and PANet	CPANSPP	Not defined
Head	3 prediction heads(52 convolutions with skip connections)	YOLOv3	YOLOv4	Multi-headed; For; Final output: Lead head Middle layers: Auxiliary Head
Backbone	Darknet-53	CSPDarknet53	CSP-P5/6/7	E-ELAN
Loss Function	Binary Cross Entropy	Binary Cross Entropy+	Binary Cross Entropy++	Binary Cross Entropy++ + Bof

3.1 How does YOLO work and how can training results be evaluated?

We already mentioned above, YOLO can be examined in 3 categories. One of them is Bounding box regression and respectively residual blocks, and intersection over union(IOU). Moreover, to evaluate the training results, we uses some metrics and methods and these will be discussed.

3.1.1 Residual blocks

The image is divided into grids of SxS dimensions and they have the same dimensions. Each grid of the picture recognizes items independently. For example, if there is a centre of items, this grid is in charge of detecting that thing. In this example, we have 3x3 grids so S=3. In addition,each cell can predict B bounding boxes. The mean of this, if one cell's B value is 2, in this cell, it can be covered maximum 2 object with bounding boxes. However, because of the limitations of YOLO, in YOLOv1 we can predict maximum 1 bounding box so B is maximum 1 for every cell. However, after YOLOv1, other new yolo versions can predict maximum 3 bounding box. So B is maximum 3 for every cell.

Figure 4

In this example, the car detected has a midpoint with green color. This point in 2x2 cell and it has a some parameter that are identified it.

3.1.2 Bounding box

The bounding box represents the detected object. When an object is detected, it is surrounded by a bounding box. The center coordinates of the detected picture, the height and breadth of the bounding box, and finally the class and probability of the detected image are all bounding box properties. These are denoted by formulas.

$$Y = (pc, bx, by, bh, bw, c)$$

To specify the parameters that are represented in the formula, YOLO needs just one bounding box regression.

The means of the parameters of formula;

- Pc: Probability of class. The value of this is between 0 and 1.0. İt defines
- Bx: x value of object midpoint and this is between 0 and 1.
- By: y value of object midpoint and this is between 0 and 1.
- Bh: height of objects. So it can be greater than 1.0.
- Bw: width of objects and also this can be greater than 1.0.
- C: class names. İt is defined by c: $(c1, c2, c3, c4, c5, .. number\ of\ class)$

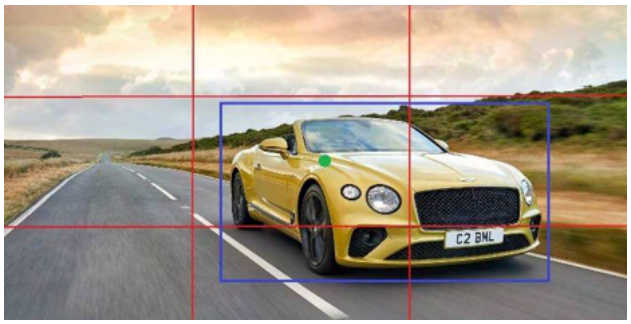


Figure 5

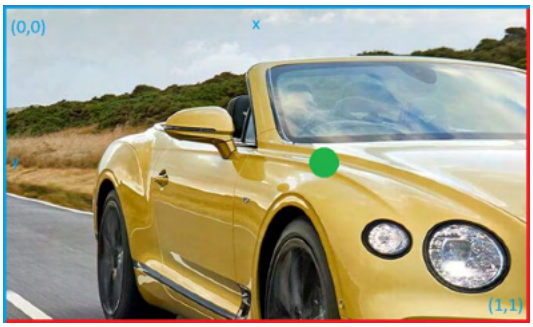


Figure 6

In this example, if we make predictions about the values, we can write the formula like this.

$$y = (0.98, 0.65, 0.50, 1.2, 1.6, car, 0,0,0,0 \dots)$$

0 means that other class is not represented the detected object. Out prediction is car with 0.98 probability.

In addition, the other likelihood is detecting multiple objects. We have objects that are midpoint in same cells. As an example fort this situation;

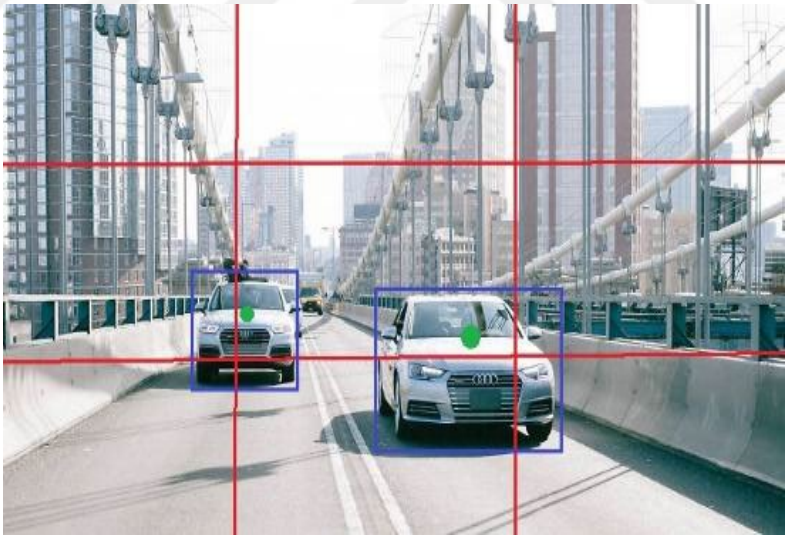


Figure 7

$Y1(pc1, bx1, by1, bh1, bw1, c)$ $Y1(0.95,0.05,0.65,0.4,0.45, car, 0,0,0,0 \dots)$

$Y2(pc2, bx2, by2, bh2, bw2, c)$ $Y2(0.99,0.80,0.85,0.7,0.85, car, 0,0,0,0 \dots)$

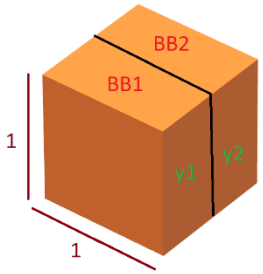


Figure 8

$$SxSx(5B + C)$$

- B: this is our bounding box value for each cell. It can be maximum 3 that we have already mentioned about it. The reason why we multiply this value with 5 is that we have pc, bx, by, bh, bw.
- C: we use C value to represent classes that we have.

To clarify this formula, for example, we have a dataset that has 20 classes. And we want to split our input image to 19 that means $S: 19$. And our B values is 3. So, $19x19x(5x3 + 20) = 396$. İn this case, our depth that means attributes is 396.

3.1.3 IOU and Map

When mAP is computed or when Non-Max Suppression (NMS) [8] is desired, intersection over union (IoU) [9]is used. It is a figure between 0 and 1 that reflects the amount of overlap between the anticipated and ground truth bounding boxes for a probability score. This step is used to identify and discard overlapping bounding boxes, which we accomplish using NMS and the IOU values. Additionally, the accuracy of a collection of object detections from a model is quantified in terms of mean average precision when compared to ground-truth objects in a dataset (mAP).There is no doubt that this is really important thing that we need to pay attention because it gives us how much we predict a object correctly.

- IoU of 0 indicates that the boxes do not overlap.
- IoU of 1 indicates that the boxes' union is equal to their overlap, suggesting that they are totally overlapping.

To calculate IOU, we need to have Ground Truth and Predicted bounding box. The required, targeted values in the test dataset are called ground truth.

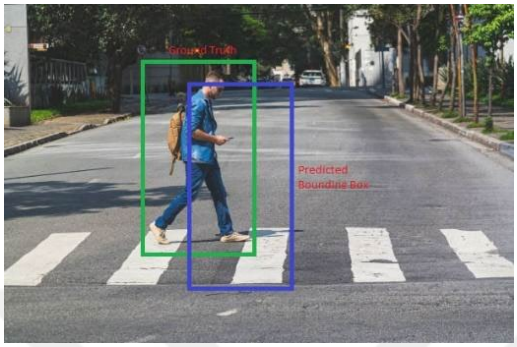


Figure 10

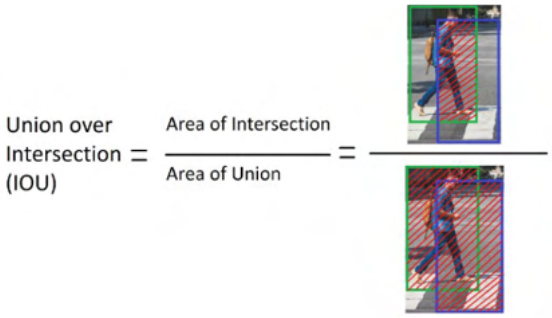


Figure 9

After finding IOU, we can apply NMS. This is a important part that help us to pick biggest IOU value to detect object in a correct way. In this step, we have some steps.

1. As the first step of NMS, we arrange the boxes in descending order of confidence.
2. Following that, we construct a confidence level. Any box with a confidence level less than this will be removed.
3. Because the boxes are ordered in decreasing order of confidence, we know that the first box on the list has the highest confidence. This first box is deleted from the list and replaced with another.
4. At this point, we'll specify an extra IOU threshold. This criterion is used to exclude boxes with a significant degree of overlap. The rationale is as follows: If two boxes overlap significantly and are both members of the same class, it is extremely probable that both boxes cover the same item. We attempt to delete the box with the lowest IOU confidence, since the aim is to have a single box for each item.
5. Repeating this procedure until all of the boxes on the list have been checked, we can pick the most accurate bounding box.

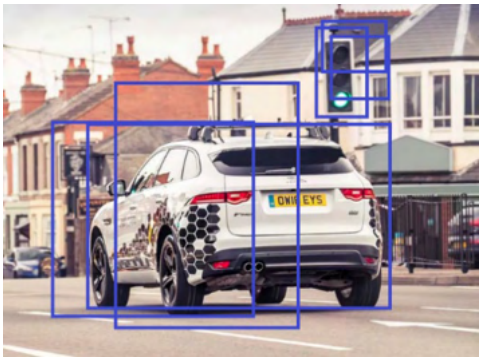


Figure 12

Figure 11

To clarify the connection between mAP, IOU, we firstly find a IOU for a bounding box. This is our a prediction for a object. We will generally cover a object with bounding boxes as a prediction. In addition, we set a threshold value. For example, if we set this values as 0.5 that is general value, our prediction should be greater than 0.5 to be true positive. Otherwise, it should be False positive.True positive(TP) values means that Predicted as positive as it turned out to be. On the other hand, False positive(FP)means that Predicted as negative as it turned out to be. With this value that are TP and FP, we can find Precision.Precision is a measure that indicates the accuracy of your forecasts. In other words, the proportion of correct forecasts.

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

Also we have Recall measure. It can be explained by how well we identified all the positives. To identify this term, we have also False Negative(FN) value that means Missed Detections.

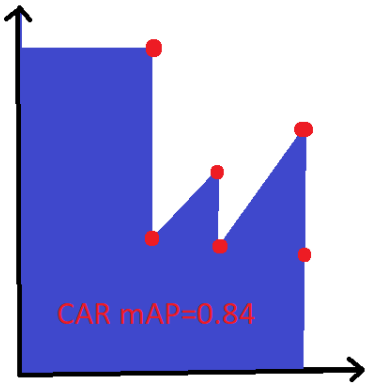
$$Recall = \frac{TP}{TP + FN}$$

- TP (True Positive): positive samples are predicted to positive correctly.
- FN (False Positive): positive samples are predicted to negative wrongly.
- FP (Fasle Negative): negative samples are predicted to postive wrongly.
- TN (True Negative): negative samples are predicted to negative correctly.

To illustrate , we have a car in an image and have 3 predicted bounding box. One of them are greater than the IOU threshold that we set and this bounding box cross with ground truth bounding box So our TP is 1. In addition, we have another bounding box

that it again cross with ground truth bounding box but IOU threshold of this is less than a value that we set. So FP is 1. On the other hand, we have last bounding box that is not cross ground truth so it is a missed detections. Because of this, FN is 1.

So, we can calculate the Precision and Recall. However, for example, we need to pay attention that if we have 30 car images, we need to make this calculations 30 times. So, we will have 30 precision and recall values for one class that we assume this as a car.



$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

Figure 13

3.1.4 F1-Score

First of all, we need to remember the values of Precision and Recall that we mentioned above. Using these two concepts we can obtain the F1-score. We need to mention in which case we will need to make use of Precision or in which case Recall. This helps us measure the performance of our model in both terms, but not in the same way. It allows us to measure the size of the error caused by Precision FPs, but helps us to measure the size of the error caused by Recall FNs. To decide which of these two metrics to use, we can give an example from the following table.

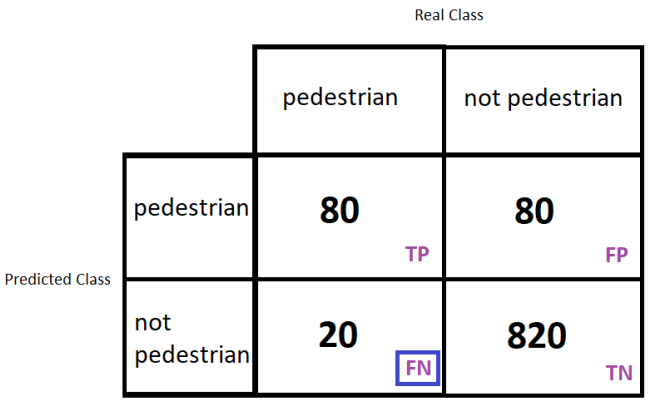


Figure 14

We can give an example on the probability of being pedestrian or not in the table. In this table, FP and FN situations are undesirable. 2. Our metric is FP. Our model states that 80 out of 900 data that are not normally pedestrians are pedestrians. Our third metric is FN. Here, our model states that 20 people who are pedestrians are not pedestrians. As can be clearly understood, Case 3 is more undesirable. Because if an autonomous vehicle does not see a pedestrian crossing a pedestrian crossing, it may hit that pedestrian. So a model should have as little FN as possible. So, it would be more logical to use our Recall metric rather than Precision. However, in what situations should the Precision metric be used. For all observations that the model predicts positively, we want most of them to be positive. Thus, for example, we expect this metric to have as accurate a degree as possible in order to infer whether it is a car or a pedestrian. So we want our model to be as precise as possible. In these cases precision is more important. But we might want to consider a situation where errors caused by FPs and FNs are almost equally undesirable. In short, it is desirable for a model to have as few FPs and FNs as possible. So we may want to maximize our precision and Recall metrics. In a normal situation, due to the trade-off between these two metrics, it is not possible to maximize both. We use the F1-score to compare which one is better according to the Precision and Recall metrics. In short, F1-score means taking the harmonic average of the Precision and Recall metrics. Precision and Recall are combined into a single point and we can formulate it as follows.

$$F1 - score = 2\,x\,\frac{Precision\ x\ Recall}{Precision + Recall}$$

$$F1 - score = \frac{2}{\left(\frac{1}{Precision}\right) + \left(\frac{1}{Recall}\right)}$$

As can be seen, both Precision and Recall are taken into account in the formula. And as it turns out, the higher the Precision and Recall together, the bigger the F1-score. F1-score can take a value between 0 and 1. 0 means the lowest score and 1 means the highest score, that is, the best score.

Table 10

	Precision	Recall	F1-score
Algorithm-1	0.50	080	0.62
Algorithm-2	0.25	0.90	0.39
Algorithm-3	0.70	0.40	0.51

It can refer to 3 cases from the table above;

- Algorithm-3 will be the right choice for us if we want the errors from FPs to be less.
- If we want less errors from FNs, Algorithm-2 will be the right option for us.
- Algorithm-1 will be the right option for us if we want the errors from FP and FN to be proportional to both metrics.

3.1.5 Mean Squared Error (MSE)

The loss function is a unit of measure that helps us to understand the extent to which we can predict the desired reality. In the loss function we have two components as inputs. These are the estimated and ground truth values. The output of the loss function is called loss, which is a measure of how well a model performs in predicting the desired outcome. The lower the loss value, the better our model is. A loss function must be determined specifically for each model. One of the most preferred functions among the loss functions is the Mean Squared Error (MSE). Calculating the MSE, we take the difference between the predictions our model makes and the ground truth, then square it and average it across the entire dataset. MSE cannot be negative and can be formulated as follows.

$$MSE = (1/N)\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$N: Number\ of\ testing\ samples$$

3.1.6 Binary Cross Entropy

Binary classification is a problem in which we need to separate our observations made in any of the two labels according to their properties. As an example, we can give the example of vehicles and pedestrians. Our goal is to place pedestrians and vehicles in two separate clusters. To overcome this problem, we use binary classification. Suppose we can recognize pedestrians and vehicles well. But to understand how well we describe them, we need a loss function, as we mentioned earlier. Thanks to the loss function, we can understand how well our model can describe the classes we have. We have previously stated that the less the difference between our Actual Class Output value and the estimation of our model, the less loss we will have. We can formulate it as follows.

$$Loss = abs(Predicted - Ground\ Truth)$$

Based on this value, we can update our model until we get the best result. Since we are talking about the concept of Binary classification, we can talk about Binary Cross Entropy, also called log loss. This is the most used loss function in binary classification problems. Binary Cross Entropy compares each of the odds predicted by our model with the Actual Class Output data, respectively. It then calculates a score based on the distance from the Actual Class Output value to the estimated value. In other words, it allows us to determine how far or close we are to our Actual Class Output value. Another term is adjusted probabilities. The definition of the probability that an observed observation belongs to the original class. If we exemplify this;

Table 11

ID	Actual Class Output	Predicted Probability	Corrected Probability
1	1	0.78	0.78
2	0	0.10	0.90

ID-1 has 1 Actual Class Output value. Therefore, Predicted Probability and Corrected Probability values are equal to each other. But ID-2 has 0 Actual Class Output. The probability of the ID-2 having an Actual Class Output value of 1 is 0.10, while the probability of having an Actual Class Output value of 0 is 0.90, which is called Corrected Probability. Corrected Probability is calculated for each sample and then Log values are calculated. The log value serves to present the small differences between Predicted Probability and Corrected Probability with less penalty. If the difference between them is high, the penalty rate will be high.

Table 12

ID	Actual Class Output	Predicted Probability	Corrected Probability	Log
1	1	0.78	0.78	-0.107
2	0	0.10	0.90	-0.045

The following formula is used to compensate for the negative values obtained as positive.

$$-\frac{1}{N}\sum_{i=1}^{N}\log(p_i)$$

And so we can express each Log value positively with this formula. As we have already stated, our Log loss value becomes our Binary Cross Entropy value. Instead of calculating the corrected probabilities, we can directly find our Log loss value with the formula below.

$$Log\ Loss = \frac{1}{N}\sum_{i=1}^{N} -(y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i))$$

$$p_i : Actual\ Class\ Output\ 1\ probability$$
$$1 - p_i : Actual\ Class\ Output\ \text{probability}$$

This way we can calculate Binary Cross Entropy. But if we are dealing with solving a multiclass classification problem (as in this project), we can formulate Binary Cross Entropy for multiclassification as follows.

$$Log\ Loss\ = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij}\log(p_{ij})$$

$$N : Number\ of\ Rows$$
$$M : Number\ of\ Classes$$

3.1.7 Categorical Cross entropy

Categorical Crossentropy is a loss function used in classification tasks of multi-class datasets. In short, it is the task of determining whether a data can belong to only one of many classes and which algorithm is used to decide. Below we can see the mathematical formula of Categorical Crossentropy.

$$Loss = -\sum_{i=1}^{\substack{Output\\Size}} y_i log\ \hat{y}_i$$

$$yi : Target\ Value$$
$$\hat{y}_i : i - th\ scaler\ value\ in\ the\ output\ value$$

3.2 Scalled-YOLOv4

After various YOLO versions, the other version YOLO that is scalled YOLOv4 has emerged. This YOLO version depend on YOLOv4. Various improvement can be found in its architecture

compared to base YOLOv4 and its derivatives. İt provides more accuracy and speed. As the name suggests, this is a scaled version of YOLOv4 with important developments and enhancements. This scaling is applied to layers sizes. Scaled YOLOv4 got YOLOv4-tiny-> YOLOv4-CSP -> P5 -> P6 -> P7 networks by using ideal network scaling techniques. Scaling object recognition models is possible by increasing the resolution of the input image, scaling the breadth of the CNN model layer, scaling the depth of convolutional layers, and scaling all of these factors simultaneously.So, we can say that our modified YOLO version has also different choice to get more speed or accuracy. P7 is most accurate choice but it is also slowest. P5/P6/P7 are developed from YOLOv4-Large that is a derivative of standard YOLOv4. Before we move on to a detailed review of its architecture and its structure, we need to understand what model scaling is.

3.2.1 The Architecture of Scalled-YOLOv4

The term "model scaling" refers to the process of increasing the number of layers in a model and so adopting a deeper network.An example of this method is the well-known and used VGG architecture. This architecture takes number values such as VGG16, VGG19 [10] according to the number of layers. To implement this method, they changed the kernel size. It is possible to come across many models developed with this method. To give an example from myself, I was able to develop Alexnet with this method and increase the accuracy ratio of standard alexnet from 46.07% to 70.03% in the tests I performed with the same parameters and dataset. Since the development of the architecture here may also vary according to the datasets to be used, each architecture lacks the same performance with each dataset. YOLO and its derivatives usually present their improvements over the COCO dataset.

If we continue with the subject of model scaling, we should also mention the following. For example, Darknet53, which is the basic backbone of YOLO, uses high input resolution so that various fine-tuning can be applied to it, thus achieving a higher assurance score, namely accuracy. In addition, when the scaling study of the model planned to be used is completed, the other important thing is to change the number of various parameters and quantitative factors. While performing these two operations, computation is taken into account and the model is scaled according to the targeted systems. Since scaling is accomplished by decreasing or increasing tiers, if we want to improve scaling in terms of speed performance, we need to scale down by decreasing the number of tiers. Moreover, it is not totally like this. We will look at this in detail. However, it is vital for autonomous vehicles to properly understand environmental factors. For this reason, this report will select a Scalled-YOLOv4 version that takes into account the balance of speed and performance. In other words, a scaled architecture will be adopted for systems with neither high-level nor low-level graphics processing units.

However, due to the importance of higher accuracy, we will be more inclined to use architec-

tures that require powerful GPUs. Considering the various scaling parameters of the architecture, a combination of the most logical of these parameters should be created. To adjust these scaling factors, we need to concentrate on the input size, backbone, and neck parts of our architecture. Here, for input size, we simply need to extend the size so that we can create a deeper architecture. For the backbone and neck parts, we apply scaling factors to the width, depth and stage parts. When we want to detect large objects, if we increase the input size, our detection accuracy will decrease. To overcome this, we must increase the depth and number of stages. Therefore, when adding upscaling, scaling is performed for the input size and stage to get the most balanced results. These two scaling steps are done first, and then the necessary scaling process for depth and width is applied according to the need.

Scaled-YOLOv4 actually has a scaled version of 3 different categories in total. These are CSP-sized YOLOv4, YOLOv4-tiny, and YOLOv-large. Since the categories, we will be interested in here are CSP-sized YOLOv4 and YOLOv4-large, we can examine these parts in detail.

3.2.2 CSP-sized YOLOv4

Let's start with the CSP-sized YOLOv4 first. Normal YOLOv4 is suitable for use with general graphics cards. The CSP-sized YOLOv4 has been redesigned to achieve the best balance of speed and accuracy. We can examine this architecture as backbone, neck and SPP. Normally, the cross-stage process of YOLOv4 is removed from residual blocks.

If we examine it more deeply, we can state the following. The capacity of the darknet layer to perform operations is specified as $k * \left[conv\left(1 \times 1, \frac{b}{2}\right) \rightarrow conv(3 \times 3, b)\right]$. The b parameter here denotes the layer channel, while k denotes the number of layers. When we want to convert a normal darknet layer to a CSP-Darknet layer, we need to scale the input size, depth (ie the number of layers) and width (ie the number of channels). Here, if we define the scaling factors we have determined as a, b and c, and our 3 main expressions from the previous sentence, respectively, we will see a table like this.

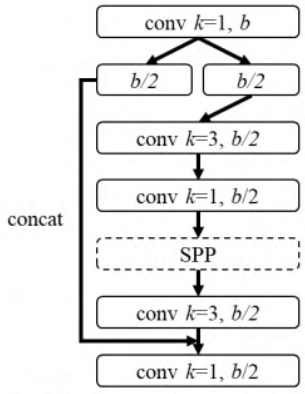
Table 13

	Default - One Layer Computation Limit	$size\ a$	$depth\ b$	$width\ c$
Darknet	$d = 5whkb^2$	$a^2 * d$	$b.d$	$c^2 * d$
CSP-Darknet	$whb^2 * \left(\frac{9}{4} + \frac{3}{4} + \frac{5k}{2}\right)$			

From this, we can deduce that CSP-Darknet can perform better than Darknet if the k value of our layer number is greater than 1. Also, the number of residual layers is 1-2-8-8-4 per stage. Subsequently, although we have converted CSP-Darknet's layers to CSP layers, the first layer is converted to Darknet's standard Darknet residual layer for better performance. Now let's focus on the neck part. Since our aim is the optimum speed and accuracy, the PAN architecture is added to the CSP architecture and our computation amount is reduced. The SPP part has taken its place in the middle layers of the neck parts, which is the first calculation group. After these improvements, we can define the name of the neck architecture as CPANSPP. We can observe this architecture and the total number of parameters, FLOPS and AP values of the overall architecture from the table and picture below.

Table 14

Backbone	Neck	Activation Function	FLOP	Total Param	AP
CD53	CPANSPP	Mish	109B	53M	47.50%



(b) reversed CSP dark layers (SPP)

Figure 15

Although YOLOv4-large is actually more suitable for object detection with a strong GPU power, it should be evaluated in this part. Because in this project, it may be possible to change these architectures according to our current system and make them suitable for use. CSP YOLOv4-large is further divided into 3 categories, YOLOv4-P5, YOLOv4-P6, and YOLOv4-P7, according to the degree of scaling. Each of these versions is based on CSP-sized and gradually scaled up to take their names. Here, although P7 gives the best accuracy value, it, unfortunately, underperforms in terms of speed. Apart from the fact that the number of deep layers of YOLOv-large and derivatives is different, the other thing we should mention is this. The depth scale of each stage is set to 2^{ds}

and the ds value for each stage is set to $[1, 3, 15, 15, 7, 7, 7]$.

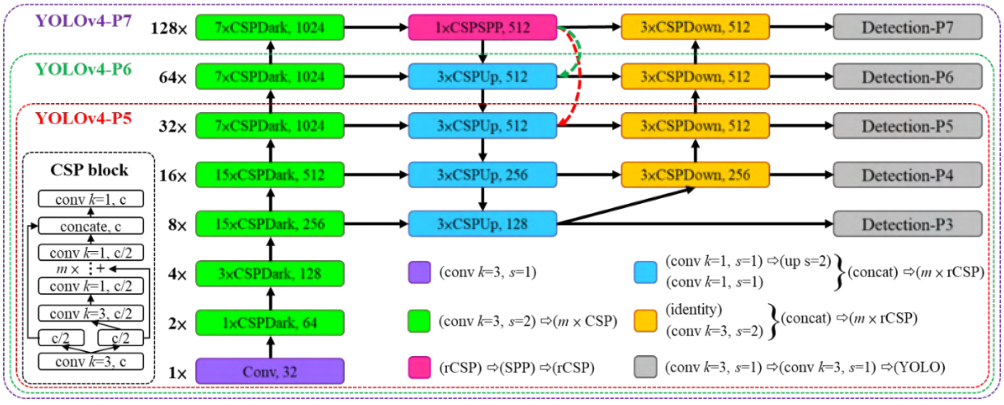


Figure 16

We can see that the difference between scalled YOLOv4-P5/P6/P7 architectures.

Now, we will look at other improvements of the architecture as a summary and more detail.The improvements and the architecture structure of scalled YOLOv4 is listed as bellow;

1. The backbone of Scalled YOLOv4 is optimized and the neck which is consisted of PAN uses Mish activation function and Cross-stage-partial(CSP). These are can be explained like;
    a. Mish can be represented non-monotonic activation function that is smooth, continuous, and self-regularized.The reason of using this activation function is that it presents a more accurate performance in object detection, unlike the well-known RELU activation function.And the formula of it is like this;

$$f(x) = x \cdot tanh\left(\varsigma(x)\right) where, \varsigma(x) = ln(1 + e^{x})$$

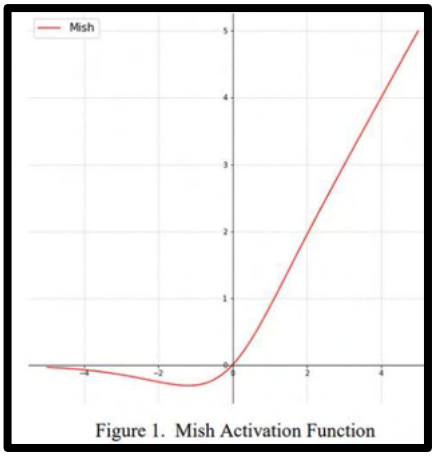


Figure 17

b. CSP connections are quick and easy to set up, and they can be used with any neural network. The purpose is; the main route carries half of the output signal. Thus, with a broad receiving field, it creates more semantic information.

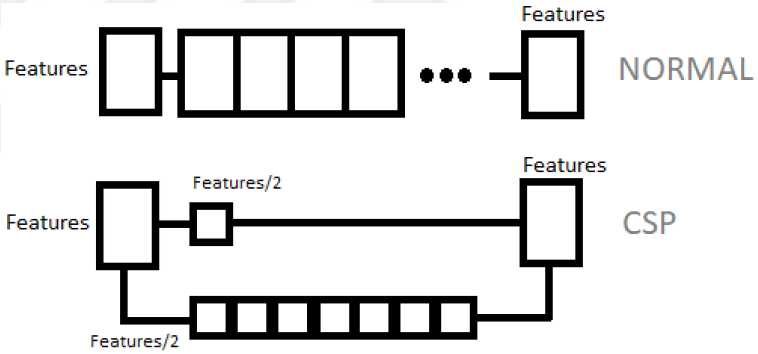


Figure 18

2. Exponential Moving Average (EMA)[11] is accompanied during training. EMA is a sort of Moving Average in which the most current data points are given greater weight than those from the past. To put it another way, it's like prioritizing the most recent experience or memory above previous ones, presuming they're represented by data points.

$$\underline{y}_{T+1|T=}\alpha y_T+\alpha(1-\alpha)y_{T-1}+\alpha(1-\alpha)^2y_{T-2}+\cdots,$$

3. The other special case for scaled YOLOv4 is the training stage for the resolution of networks. In this case, while just one neural network is trained for all resolutions for YOLOv4, a different neural network is trained for every resolution of the network in

scaled YOLOv4.

4. Layer object classes normalizers have been tweaked and enhanced.
5. The activations for Width and Height have been altered, allowing for quicker network training.
6. The other performance enhancement was provided by changing loss functions. By changing loss function, this helps to remove grid sensitivity in the same manner as YOLOv4 does, but more forcefully for bx and by. On the other hand, in terms of bw and bh, this restricts the bounded-size box to four anchor sizes.

$$Y = (p_C, b_x, b_y, b_h, b_w, c)$$

$$b_x = \sigma(t_x) * 2 - 0.5 + c_x$$
$$b_y = \sigma(t_y) * 2 - 0.5 + c_y$$
$$b_h = (\sigma(t_h) * 2)^2 * p_h$$
$$b_w = (\sigma(t_w) * 2)^2 * p_w$$

7. While standard YOLOv4 uses DARKNET, scaled YOLOv4 is got power from Pytorch. The YOLOv4 version in Pytorch estimates better coordinates (better AP), but finds lesser objects (worse AP50). By adding CSP[13] to the neck and using Mish activation function for every layers, we can remove the lack of Pytorch. Thanks to the this implementation, better AP50,AP and FPS can be reached.

Table 15

Model	Back-bone	Test Size	AP	AP50	FPS	width scaling factor
YOLOv4	CD53	512	43%	64.90 %	83	Undefined
CSP-sized	**CD53s**	**640**	**48%**	**66.20 %**	**73**	**Undefined**
YOLOv4-P5	CSP-P5	896	51.4	69.90 %	43	Undefined
YOLOv4-P6	CSP-P6	1280	54.3	72.30 %	32	1
YOLOv4 -P6 + BoF	**CSP-P6**	**1280**	**54.4**	**72.70 %**	**30**	**1**
YOLOv4-P7	CSP-P7	1536	55.00%	72.90 %	15	1.25

3.3 YOLOv7

The YOLO-V7 algorithm is a different mainstream in the case of real-time object detectors than the real-time object detectors that are currently in use. The suggested algorithm, YOLO-V7, focuses on enhancing the procedure of training instead of some optimization of the architecture. To have better accuracy of the object detection that will improve the training cost function without enhancing the inference cost, this algorithm provides us with some optimised methods that we will consider. The proposed modules and methods of optimization are called bag-of-freebies. [14]

In the case of training and detection objects, re-parametrization and dynamic label assignment are significant subjects. After emerging these concepts, they have come with some problems. With YOLO-V7, these issues which are detected are tried to solve by devising brilliant methods to address them. With the idea of the gradient propagation path, we examine the model re-parameterization procedures that are relevant to the layers in various networks and present a planned re-parameterized model. Additionally, training a model with many output layers will create new problems with dynamic label assignment technology, which is related to assign dynamic targets for the outputs of different branches. To handle this issue, YOLO-V7 uses a new label assignment method, which is named coarse-to-fine lead guided label assignment.

With YOLO-V7, these improvements can be listed below.

- In order to significantly increase object detection accuracy while maintaining the same inference cost, a number of trainable bag-of-freebies approaches are built.
- Two additional problems for the evolution of object detection methods are discovered, which is about the re-parameterized module replacing the original module and which is related to the dynamic label assignment approach that handles the assignment to various output layers.

To handle some hardships caused by these issues, this method is suggested as below.

- Two methods called “extend” and “compound scaling” for object detectors which are able to utilize parameters and computation productively. Then, the suggested method is able to decrease the parameters by approximately 40% and in addition, 50% computation also can be reduced in the object detector. With these improvements, the YOLO-V7 can achieve better inference speed with better detection accuracy.

We can list several YOLOV7 versions which are specific to some hardware configurations. For example, in this report we will use standard YOLO-V7 instead of other its other derivatives. This

architecture, YOLO-V7, is designed for normal GPUs. However, its derivatives such as YOLO-V7-tiny are special for edge hardware. Moreover, for cloud hardware, we can talk about YOLO-V7-W6. For these architectures, various scaling options are used. However, in this paper, we will focus on how YOLO-V7 and its derivatives work so we will take a deep look at the improvements in these architectures.

For instance, standard YOLO-V7, stack scaling on neck and compound scalling techniques are used in order to actualize scaling up for both depth and width of the whole model so as to get YOLO-V7-X. Additionally, the ELAN architecture is used for YOLO-V7-E6, which is obtained from YOLOV7-W6 that uses the compound scaling method. Regarding activation function, only YOLO-V7-tiny is using leaky RELU, other than other YOLO-V7 and its derivatives use SILU activation function.

- SILU activation function:

  A continuous and "undershooting" version of the linear rectifier unit, the SiLU's activation is calculated by multiplying its input by the sigmoid function (ReLU).[15]

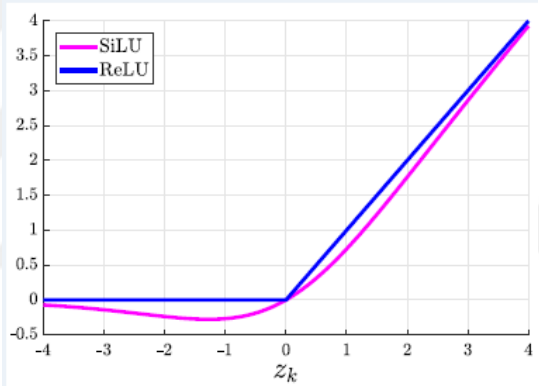


Figure 19

$$a_k(z_k) = z_k\sigma(z_k)$$

$$\sigma(x) = \frac{1}{1+e^{-x}}\ ,which\ is\ sigmoid\ function$$

With big magnitude values of z_k, SILU is nearly similar to the RELU activation function. However, SILU is not a monotonically increasing compared to RELU. The $z_k$value has a global minimum, which is nearly to -0.28 in the case of $z_k \cong -1.28$. Moreover, If the derivative is 0 which is global minimum, it helps to learn big weights by providing a soft floor.

3.3.1 Model re-parameterization

Model re-parameterization methods are known as ensemble technique, which are represented as 2 categories, “module-level ensemble” and “model-level ensemble”. To get the inference model,

two typical model-level reparameterization techniques can be listed. One of the methods is to train numerous, identical models using various training datasets, and then taking average the model weights. This kind of technique divides a module during training into numerous identical module branches, which are then combined during inference to create a single, fully equivalent module. All suggested re-parameterized module cannot be perfectly applied to various architectures. In light of this, YOLO-V7 includes a new re-parameterization module and corresponding application techniques for different architectures. Some decent techniques are.

- Linear over-parameterization to train compact convolutional networks
- ACNet: Strengthening the Kernel Skeletons for Powerful CNN via Asymmetric Convolution Blocks

3.3.2 Scaling of Model

A technique for scaling up or down an existing model is called model scaling. In order to achieve a nice balance between the number of network parameters, computation, inference speed, and accuracy, the scaling of the model method typically uses various scaling factors, such as input image size, depth, and channels' number. One of the generally used methods to scale a model is NAS(Network architecture search), which can search decent scaling factors found in search space by taking into account not to be complicated rules. However, NAS has too much computation cost to determine the model scaling factor by searching. The majority of common NAS architectures work with scaling factors that are not highly correlated and each model scaling technique examines each scaling element separately. Because of the using an architecture which is concatenation-based, these NAS architecture is not suitable to use in YOLO-V7. Therefore, YOLO-V7 scaling factor architecture is designed as a new compound scaling method.

3.3.3 Architectural Improvements

This sub section will discuss some improvements came with YOLOv7.

3.3.3.1 E-ELAN (Extended efficient layer aggregation networks)

In most studies to build an effective architecture, the main criteria are the number of parameters used, the amount of computation and density. It is aimed to realize the model scaling, to give more importance to the amount of elements that the tensors have at the outputs of the layers. Beyond such design goals, some architectures (CSPVoVNet) use gradient path analysis so that the weights of different layers can understand properties with greater density which helps in faster and more accurate inferences. In order to design a decent network architecture, the shortest gradient path must be determined, and with it, deep structure architectures can learn and converge with higher accuracy. An example of this is the ELAN architecture. However, YOLO-V7 has designed and pre-

sented the E-ELAN architecture, which is a derivative of this architecture.

A scaled version of the ELAN architecture can be said to be stable, in the case of the gradient path length and the number of computational blocks are not taken into account. However, if the number of calculating blocks increases, this stability will deteriorate and as a result, the number of parameters used will decrease. E-ELAN is assisted by expand, shuffle, and merge features to continuously increase the learning ability of the network without changing the gradient path. It is aimed to expand the channel of computation blocks by using the group convolution theorem.

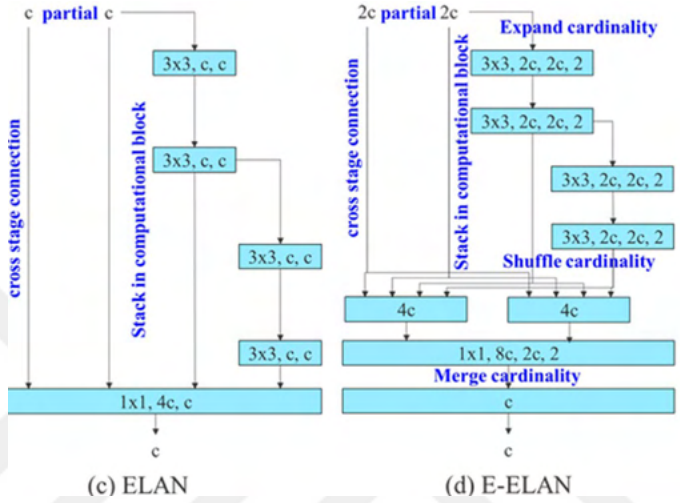


Figure 20

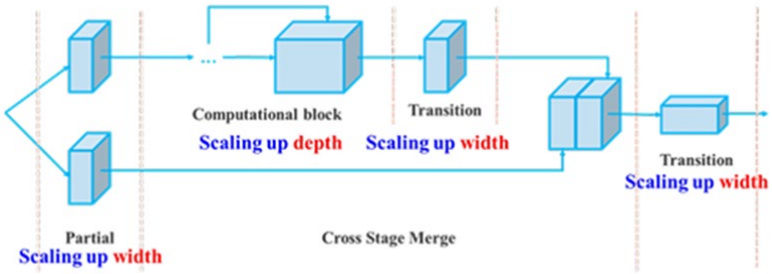
3.3.3.2 Model scalling in the case of concatenation-based models

Model scaling is mostly used to modify certain model properties and produce models at various scales to accommodate various inference speeds. As an example, the model that we used called Scalled-YOLO-V4 scales to alter the number of phases. In order to be able to independently analyze the effect of each scaling factor on the parameter amount and calculation, we can perform scale-up and down-scale, and the degrees and outer degrees of the layers do not change in the process.

When enlarging and reducing the depth with these methods, and if these methods are adapted to a concatenation-based architecture, it causes a decrease or increase in the degree of a translator layer that comes after the concatenation-based computation block. This means that we cannot analyze the different scaling factors separately. However, it should be underlined that it is necessary to consider them together. Because of this reason, compound scalling method to use in concatenation-based model should be defined.

Calculating the change in the output channel of a computational block is necessary for scaling the depth factor of that block. The transition layers will then undergo width factor scaling with the

same amount of change. During actualizing scaling on a concatenation-based model, in a computational block, just the depth needs to be scaled; the remaining transmission layer is completed with the appropriate width scaling.



(c) compound scaling up depth and width for concatenation-based model

Figure 21

To illustrate the improvement of compound scaling compared to width only and depth only scaling, the table below shows how the compound scaling method used inYOLO-V7 and its derivatives has a good performance.

Table 16

Model	Param	FLOPs	AP_{VAL}
X	47.0M	125.5G	51.7%
X with width only(x1.25)	73.4M	195.5G	52.4%
X with depth only(x2.0)	69.3M	187.6G	52.7%
X with compound	**71.3M**	**189.9G**	**52.9%**

The model X is to illustrate the improvement on the performance in the case of param flops and apval For Y dataset.

3.3.4 BoF(bag-of-freebies)

In this chapter, we will discuss some techniques such as re-parameterized convolution and Coarse for auxiliary and fine for lead loss.

3.3.4.1 Coarse for auxiliary and fine for lead loss

We can talk about the definition of the Deep supervision technique by adding the auxiliary head and lossy shallow weights to the middle layers of a mesh as a guide. Deep supervision makes significant contributions to the accuracy performance of the model.

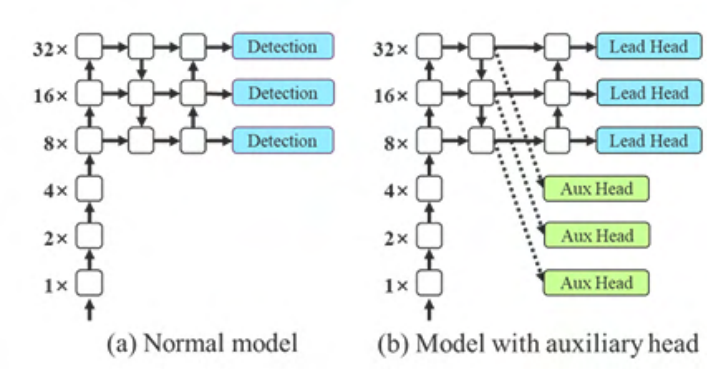


Figure 22

Another important method is label assignment. Based on YOLO, this technique is done using the IOU, and it does it for the prediction of bounding box regression and ground truth as the soft label of abjectness.

Moreover, the head that is responsible for the final output (lead head) and the head that is used to assist in training (auxiliary head) without any dependency on deep supervision must be trained.

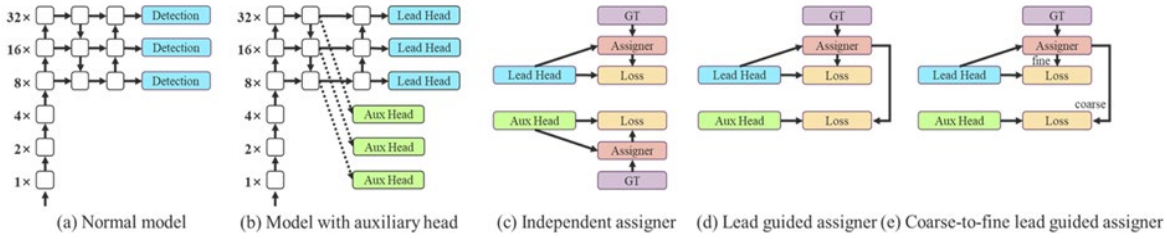


Figure 23

The technic called soft label assigner helps to assign soft labels to head that is used to assist training and to assign soft labels to head that is responsible for the final output. In some studies, this technic is used by keeping apart auxiliary head and lead head. After this process, these heads benefit their own ground truth and estimation results to actualize label assignment. In YOLO-V7, two deep supervision technics are presented, which are lead head guided label assigner and coarse-to-fine lead head guided label assigner. Respectively, first one is activated by prediction of lead head and ground truth in order to have lead and auxiliary head's labels simultaneously, whereas the second one is more complicated and related to some other dependencies so it will be explained later.

3.3.4.1.1 Lead head guided label assigner

It is primarily calculated using the lead head's prediction result and the ground truth. Finally, it produces soft label after going through the optimization procedure. For the auxiliary head and lead head, these soft labels will be influenced as a goal-based model of training because to the reason that the lead head has a ability of perfect learning. Hence, the soft label that is produced from it ought to be more indicative of the distribution and correlation between the target and the source data. With the process, the aux. head can learn the knowledge which lead head has already got, which means that the lead head can be able to concentrate on taking information being not learned yet.

3.3.4.1.2 Coarse-to-fine lead head guided label assigner

In order to construct soft label, it also employed the lead head's anticipated outcome and the ground truth. Nevertheless, as part of the procedure, two distinct sets of soft labels are created, namely coarse and fine labels, with the fine labels being identical to the soft labels created by the lead head guided label assigner. Additionally, the coarse label is produced by loosening the restrictions placed on the positive sample assignment process, which allows more grids to be classified as positive targets because aux head capability in terms of learning is not powerful compared to the lead head. In this architecture, optimizing the aux. recall head will be have more importance because of avoiding significant information. Some limitations on the decoder to reduce the influence of those extra coarse positive grids by preventing them from producing soft label perfectly are imposed because it may yield poor results before the expectations are met if the added weight of the coarse label is near to that of the fine label. With these implementations, the significance of the lead and coarse label in the case of learning process can be set with a better procedure to facilitate the process, which means that the fine label will be higher than the coarse label in terms of the upper bound.

To demonstrate the improvement on performance, we can compare independent label assignment in the case of both lead and aux head with proposed lead guided and proposed coarse-to-fine lead guided techniques. From the table, there is no doubt that proposed lead guided method shows better performance compared to independent label assignment method. Moreover, proposed coarse-to-fine lead guided method also illustrates better performance compared to proposed lead guided method in the case of AP_{VAL75}. By increasing assistance loss on any model, overall performance is increasing.

Table 17

Model	AP_{VAL75}
Independent	60.9%
Lead guided	61.0%
Coarse-to-fine lead guided	61.1%

When we look at the aux without constraint and aux with constraint on the proposed coarse-to-fine lead guided method, we can see from the table that better results can be obtained by restricting the upper bound of objectness by the distance from the object's centre.

Table 18

Model	AP_{VAL75}
Base model(V7-E6)	60.7%
Coarse-to-fine lead guided (aux without constraint)	61.0%
Coarse-to-fine lead guided (aux with constraint)	61.1%

The other improvement is partial aux head which is implemented E-ELAN architecture not to lost information in the next level pyramid coming from middle level pyramid. Multiple pyramids can be found in YOLO-V7 and aux head can connect to pyramid in middle layer. Each of the pyramid of lead head can have the information from objects in the case of different sizes by connecting aux head before merging cardinality and after a sequence of the feature map. The table below shows the performance between coarse-to-fine lead guided and its partial version techniques. There is no doubt that with coarse-to-fine lead guided(partial aux with constraint) shows better performance. This comprassion depends on V7-E6E(E-ELAN version) instead of V7-E6.

Table 19

Model	AP_{VAL75}
Base model(V7-E6E)	61.5%
Coarse-to-fine lead guided(aux with constraint)	61.6%
Coarse-to-fine lead guided(partial aux with constraint)	62.1%

3.3.4.2 Re-parameterized convolution

In YOLO-V7, Re-parameterized convolution technique is a useful method to get good performance. However, combining this method with other networks (ResNet and DenseNet) can cause problems in terms of accuracy and gradient flow propagation paths are used to determine how it can be adapted to other networks. With re-parameterized convolution, 3x3, 1x1 convolution and identity connectivity are combined in a single convolution layer. With the analyzes made, it was decided not to use the identity connection in the design of parameterized convolution architecture (RepConvN). This is due to the loss of concatenation, which provides residuals in ResNet and greater gradient diversity for different feature maps in DenseNet. Therefore, in YOLO-V7, there

should be no identity linkage any longer or when a combined convolution layer is replaced by a reparameterized convolution.

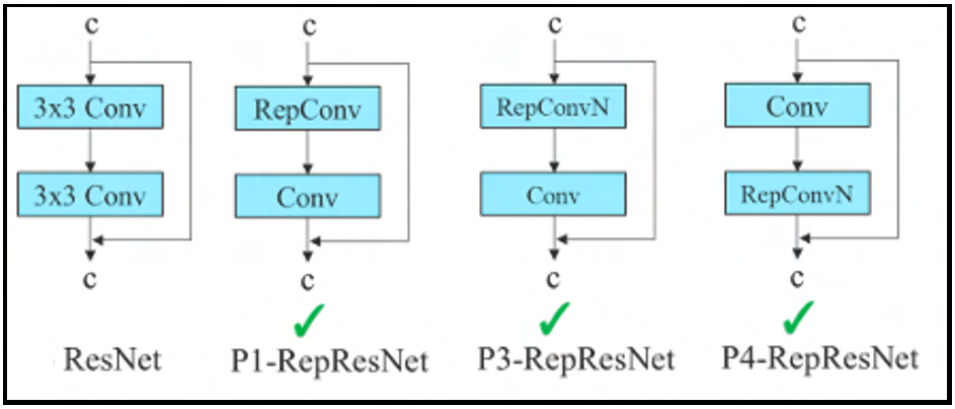
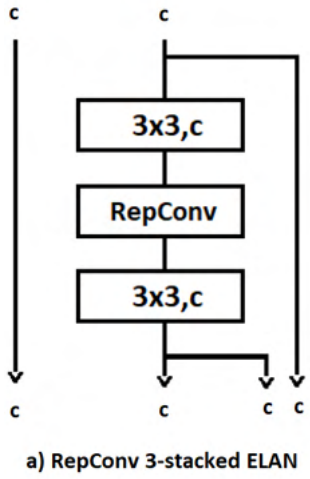


Figure 24

To illustrate the improvement of proposed re-parameterized technique, the table below shows how the re-parameterized method used in YOLO-V7 and its derivatives has a good performance. However to demonstrate these performance enhancement, first concatenation-based model and then residual-based model will be used. For the concatenation-based model, 3-stacked ELAN and for residual-based model CSPDarknet will be used. The figure below shows how the best performance can be achieved with changing 3x3 conv layers and 3-stacked ELAN with re-parameterized convolutional layers with a table showing the performance in the case of AP.

Table 20



Model	AP_{VAL}
Concatenation-based model with 3-stacked ELAN	52.26%
Best variation replacing conv with RepConv	52.33%

Figure 25

When we comes to show improvements with residual-based model which is CSPDarknet by reserving the position of 1x1 and 3x3 conv. layers to get proposed reparameterized model in YOLO-

V7 and its derivatives, we can see the figure and the table to understand how the performance changed.

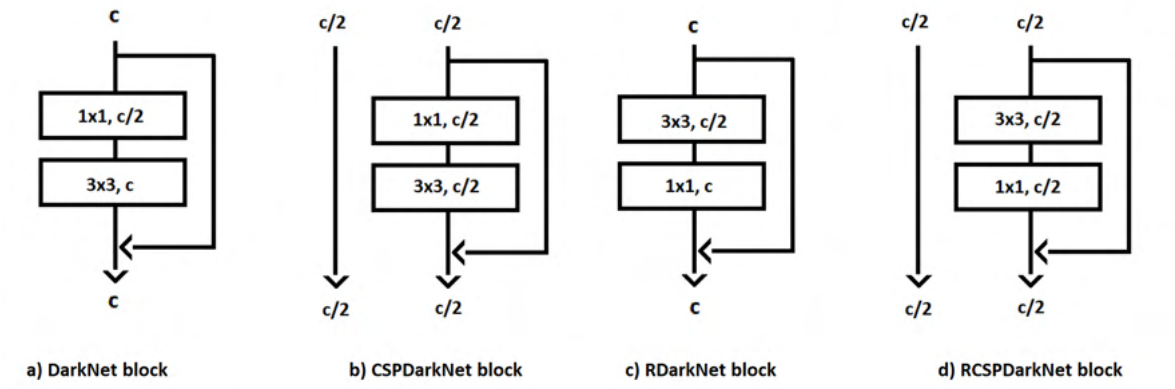


Figure 26

Table 21

Model	AP_{VAL}
CSPDarkNet	54.82%
Re-parameterized CSPDarkNet	54.67%
RCSPDarkNet	54.36%
Re-parameterized RCSPDarkNet	54.85%

4 EXPERIMENTS AND SIMULATION RESULTS

In this chapter, we will discuss training results that are actualized with some methods which are fine tunning and layer freezing. These trainings depend on Scalled-YOLOv4 and YOLOv7 with KITTI and WAYMO datasets.

Before starting, we need to look at the meaning of fine tuning and layer freezing. First of all, fine tuning means that when a neural network is fine-tuned, its weights are used as the initialization for a new model that is being trained on data from the same field (ex: images). It is employed to:

- accelerated training
- Trying to get rid of the disadvantages of small dataset

There are different approaches, such as "freezing" some of the already trained weights or training the entire initialised network (enerally entire layers).

With fine-tuning, we don’t only change the CNN architecture but also re-train it so as to get new classes if you need. On the other hand, transfer learning is a subclass of fine tuning. Transfer

learning is typically used for problems where the dataset contains insufficient data to fully train a model from beginning and we can list the steps:

- Get the layers from the model you've already trained.
- Freeze them to prevent losing any of the data they hold during upcoming training sessions.
- Stack some fresh, trainable layers over the frozen ones. They will discover how to anticipate using the previous features on a fresh dataset.
- Train the new layers using the dataset you have.
- This is the step that is fine-tuning. The process of fine-tuning entails unfreezing the whole model you received (or a portion of it) and retraining it using the fresh data at a very slow learning rate. By gradually adjusting the pretrained properties to the fresh data, this has the potential to provide significant improvements.

By the way, we already talked about the freezing process. The weights of a layer cannot be changed once it has been frozen. Transfer learning frequently employs this method, where the basic model trained on a different dataset is frozen.

Moreover, before looking at the training results, to interpret the results, we may need some parameters means found in graphs.

1. Train/box_loss : bounding box regression loss in terms of training (Mean Squared Error)
2. Train/obj_loss : The confidence of object presence is the objectness loss in terms of training (Binary Cross Entropy).

   In this section, the graph displayed by our misclassified classes over 300 epochs depending on our data. For example, if our class number was 1, then this value would form a straight line at 0.

3. Precision (We mentioned before)
4. Recall (We mentioned before)
5. Val/box_loss= bounding box regression loss in terms of validation (Mean Squared Error)
6. Val/obj_loss= The confidence of object presence is the objectness loss in terms of validation (Binary Cross Entropy).

7. mAP_0.5 : mAP_0.5' is the mean Average Precision (mAP) at IoU (Intersection over Union) threshold of 0.5.
8. mAP_0.5:0.95 : mAP_0.5:0.95' is the average mAP over different IoU thresholds, ranging from 0.5 to 0.95.

4.1 Scalled-YOLOv4 training with KITTI and WAYMO datasets

In this section we will discuss our results but firstly we need to look at all the graphs we got. In this chapter, we have scalled-YOLOv4 results with KITTI and WAYMO datasets.

4.1.1 Waymo Dataset results

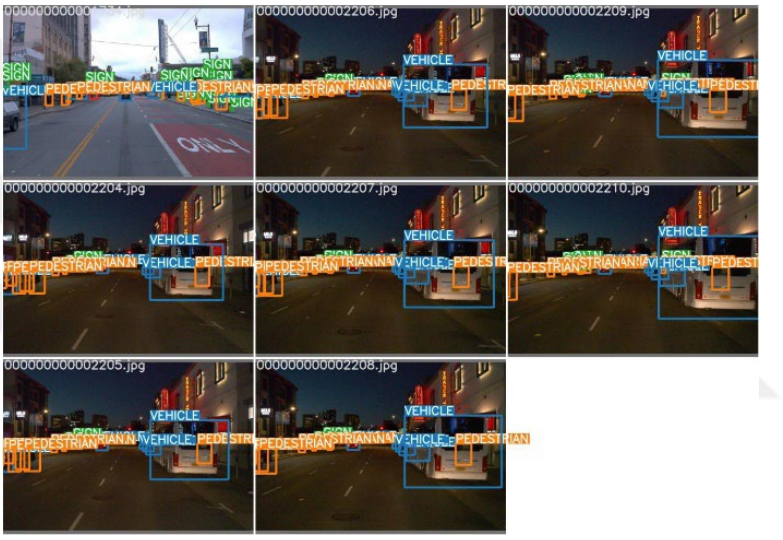


Figure 27

Table 22

Epoch	Batch_size	Img_size	Precision	Recall	mAP@.5	mAP@.5:.95	Weight
100	4	640	0.2151	0.1454	0.1697	0.08914	S-YOLOv4-csp

4.1.2 KITTI dataset results

Figure 28

Table 23

Epoch	Batch_size	Img_size	Precision	Recall	mAP@.5	mAP@.5:.95	Weight
300	4	640	0.3331	0.7133	0.5432	0.407	S-YOLOv4-csp

4.1.3 Comparison of Scalled-YOLOv4 training result

- We can easily see that for 4.1.1 waymo dataset unfourtounatelly can not perform good performance because of the reason why the WAYMO dataset we used is a small dataset. Moreover, this dataset images extracted from a video sequence. Therefore, these images are very similar each other. Because we could not use entire dataset, our dataset is not sufficient to train our algorithm. In addition, the reason why we use 100 epoch is that we trained this algorithm with this dataset before and we saw the performance does not chance if it is more than 100 epoch. So, we limited the epoch number to 100 for every WAYMO dataset traninig processes.
- There is no doubt that KITTI dataset got a good result with KITTI dataset. Because it has various types of images. Our Ap50 value is 0.5432 but the recall value is 0.71, which is important because if our FN value will increase, the recall approaches to 0. And in the autonomous cars industry, we are prone to have less FN value which is corresponding to high recall. We discussed why we need to have small FN value in previous chapters.

4.2 YOLOv7 training with KITTI and WAYMO datasets

In this section we will discuss our results but firstly we need to look at all the graphs we got. In this chapter, we have YOLOv7 results with KITTI and WAYMO datasets.

4.2.1 KITTI dataset results without finetuning and layer freezing

Figure 29

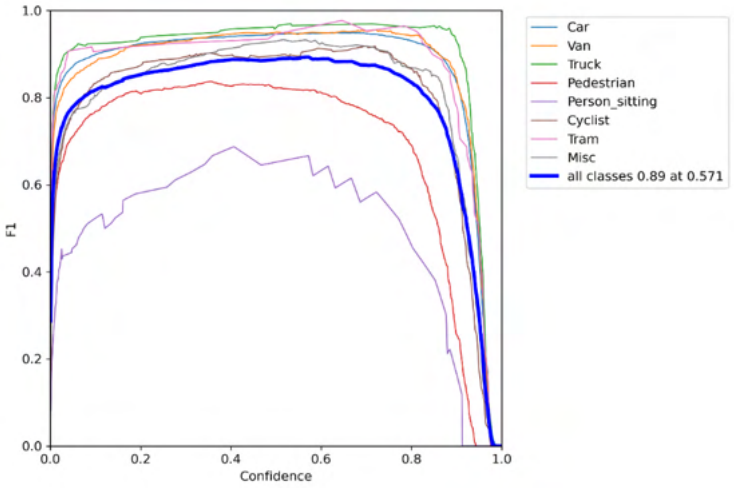


Figure 30

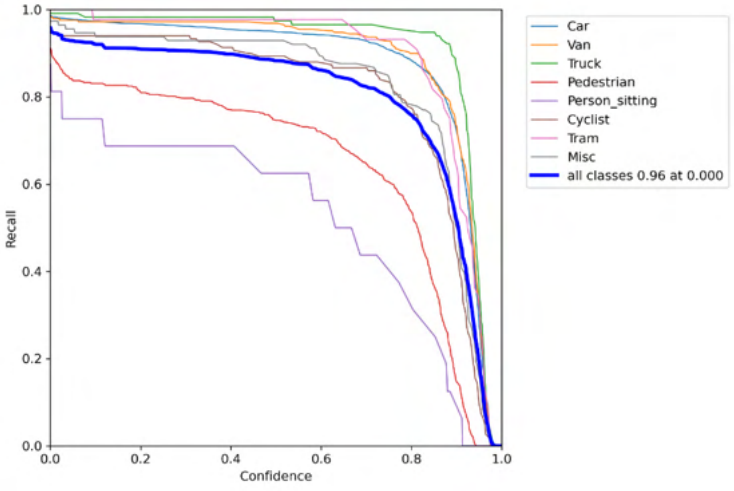


Figure 31

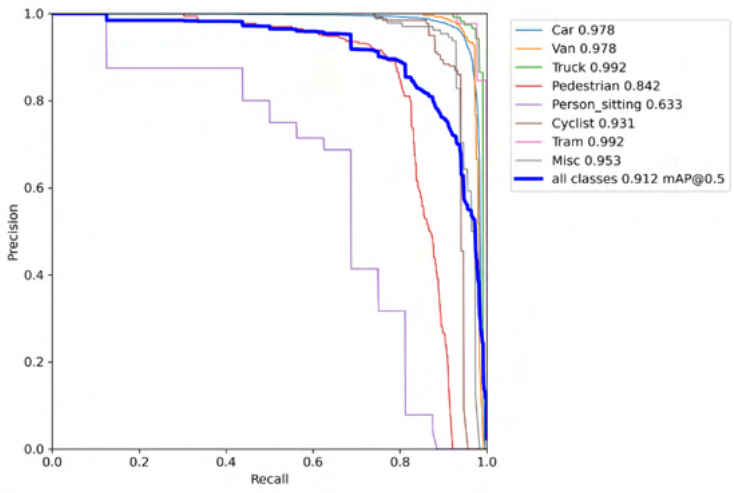


Figure 32

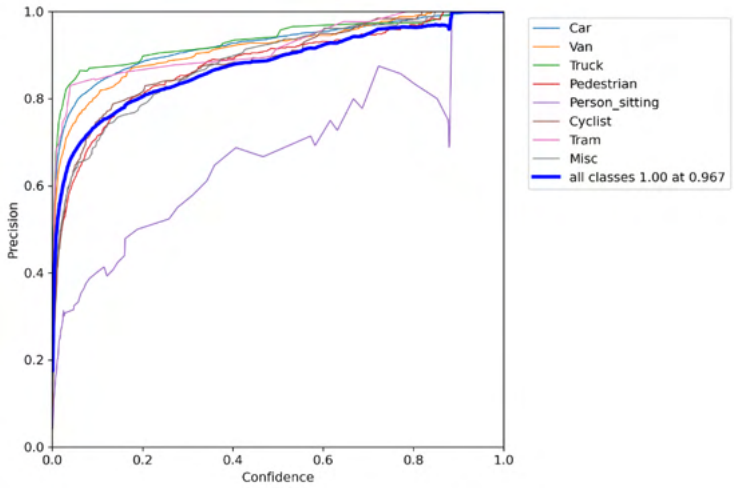


Figure 33

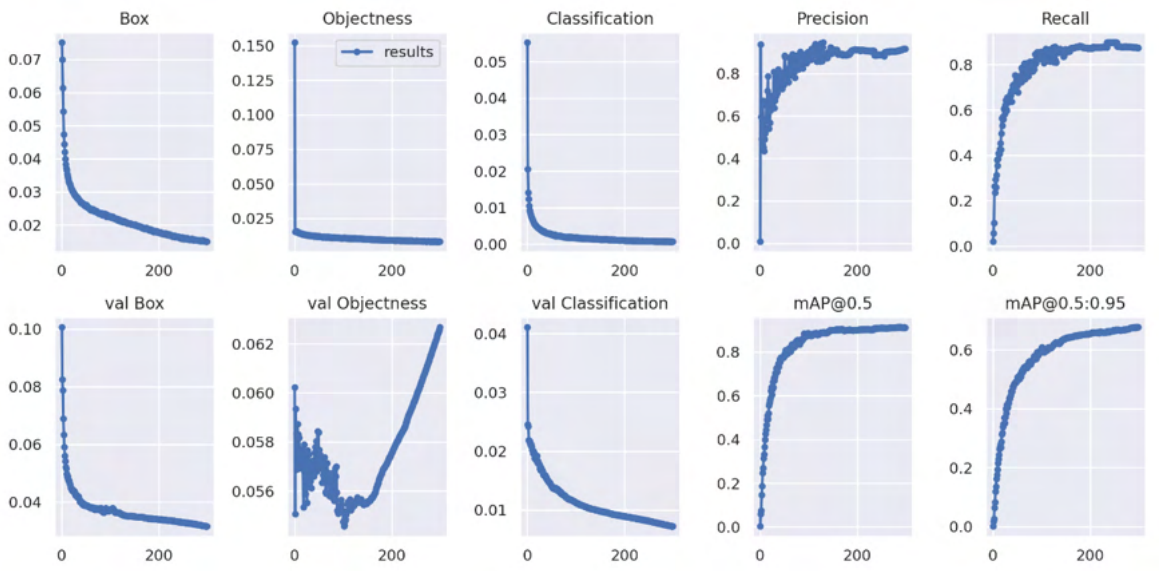


Figure 34

Table 24

Epoch	Batch_size	Img_size	Precision	Recall	mAP@.5	mAP@.5:.95	Weight
300	4	640	0.9163	0.8743	0.9122	0.6765	YOLOv7

4.2.2 KITTI dataset results with finetuning and first 50 layer freezing

Figure 35

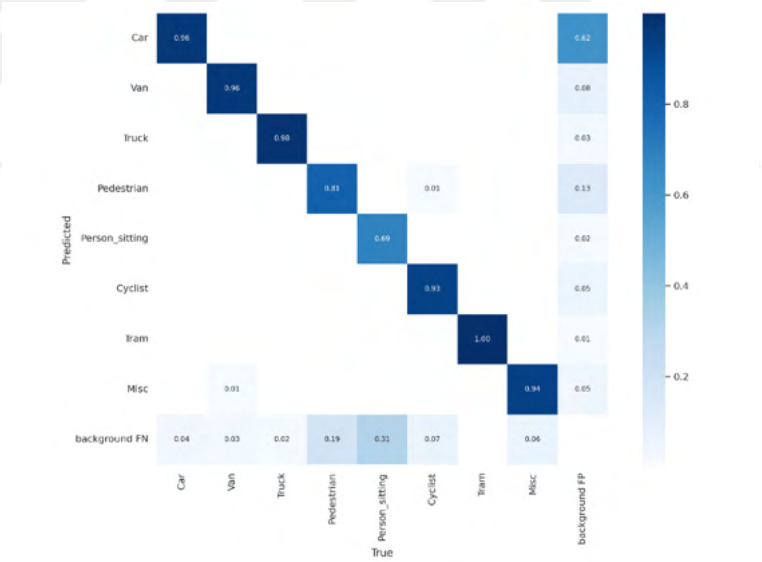


Figure 36

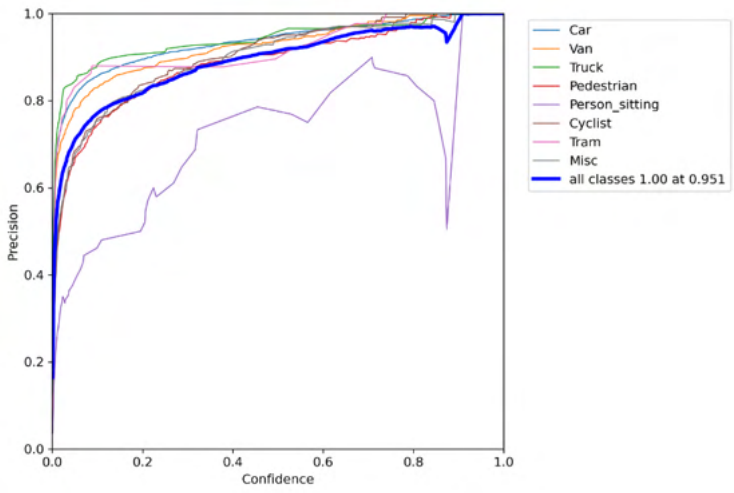


Figure 37

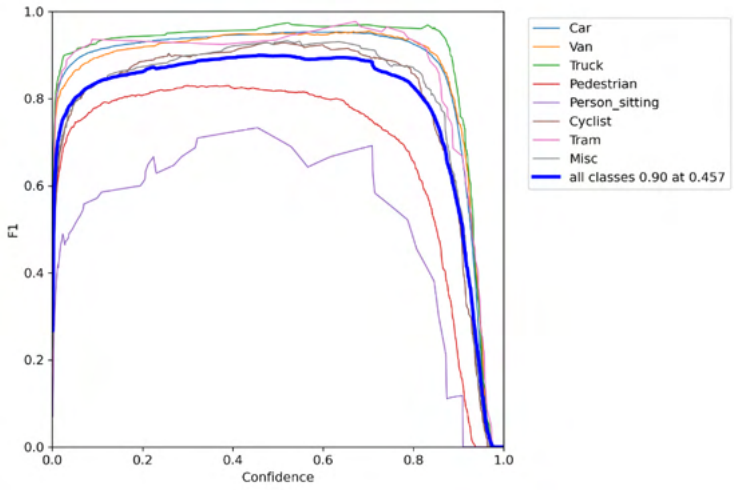


Figure 38

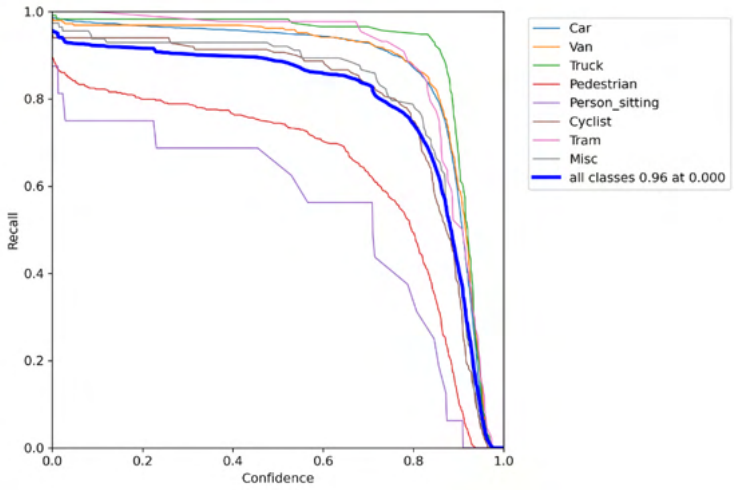


Figure 39

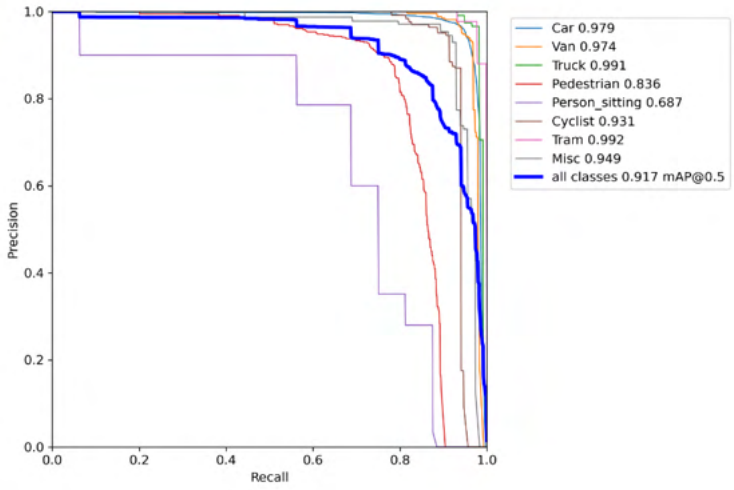


Figure 40

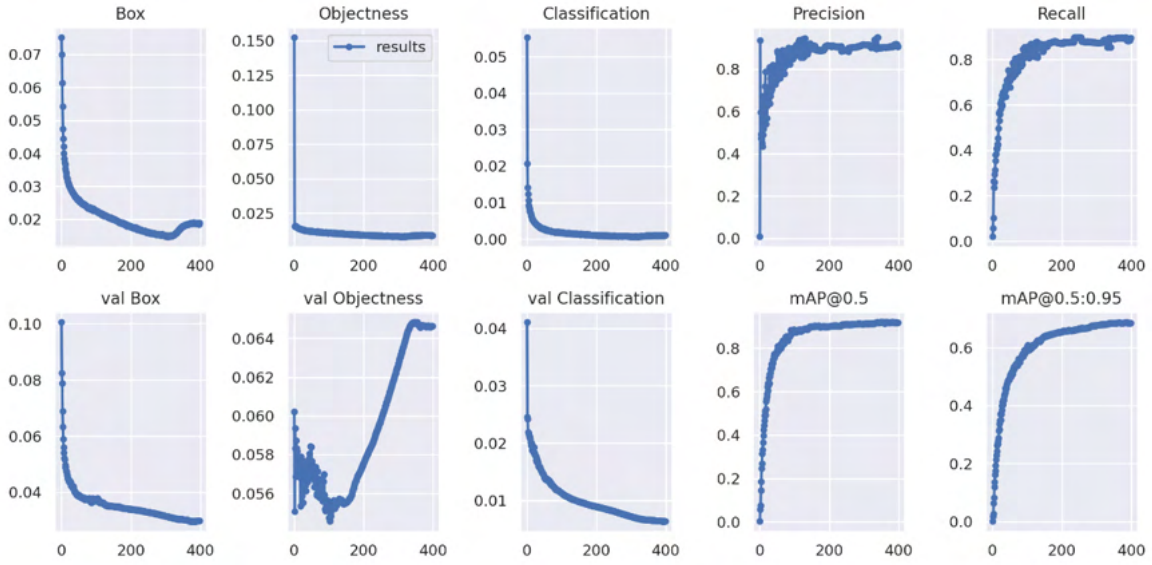


Table 25

Epoch	Batch_size	Img_size	Precision	Recall	mAP@.5	mAP@.5:.95	Weight
+100	4	640	0.9073	0.8949	0.9174	0.6852	Best_YOLOv7

4.2.3 KITTI dataset results with finetuning and first 101 layer freezing

Figure 41

Figure 42

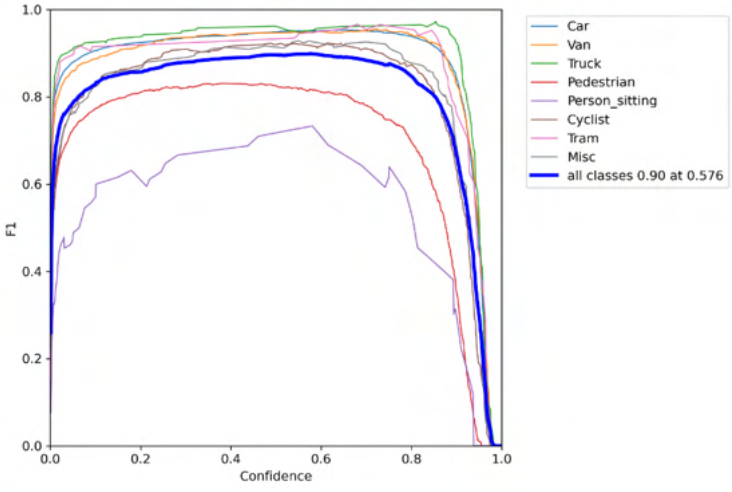


Figure 43

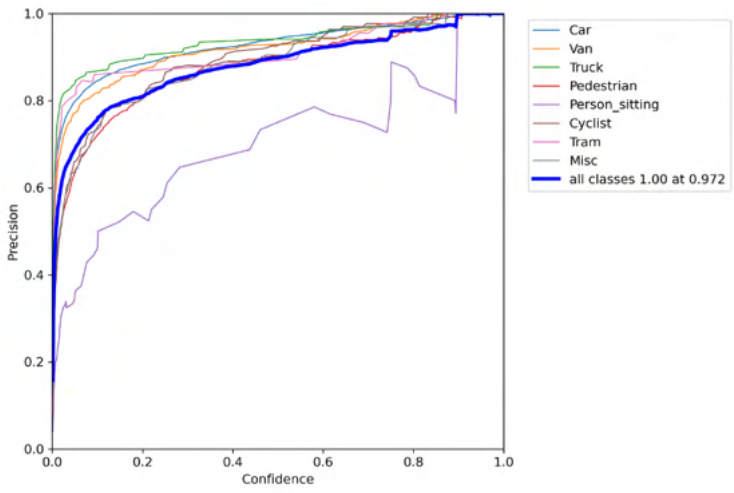


Figure 44

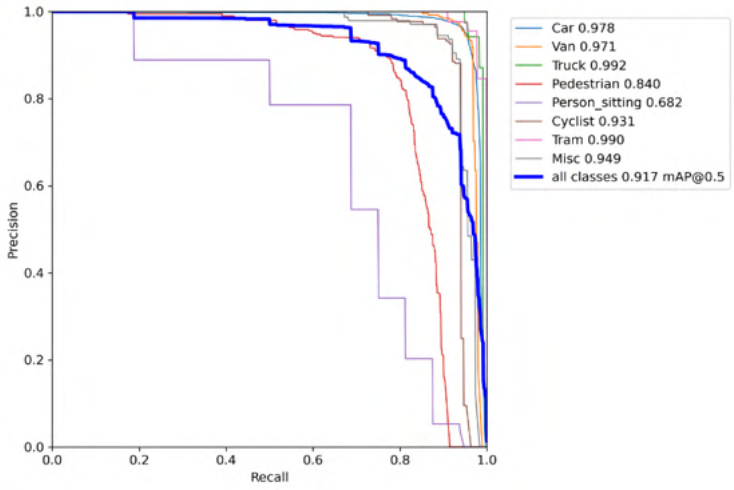


Figure 45

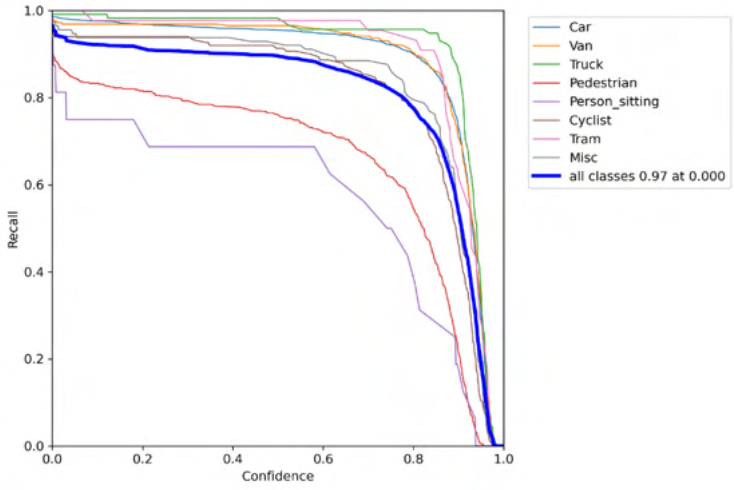


Figure 46

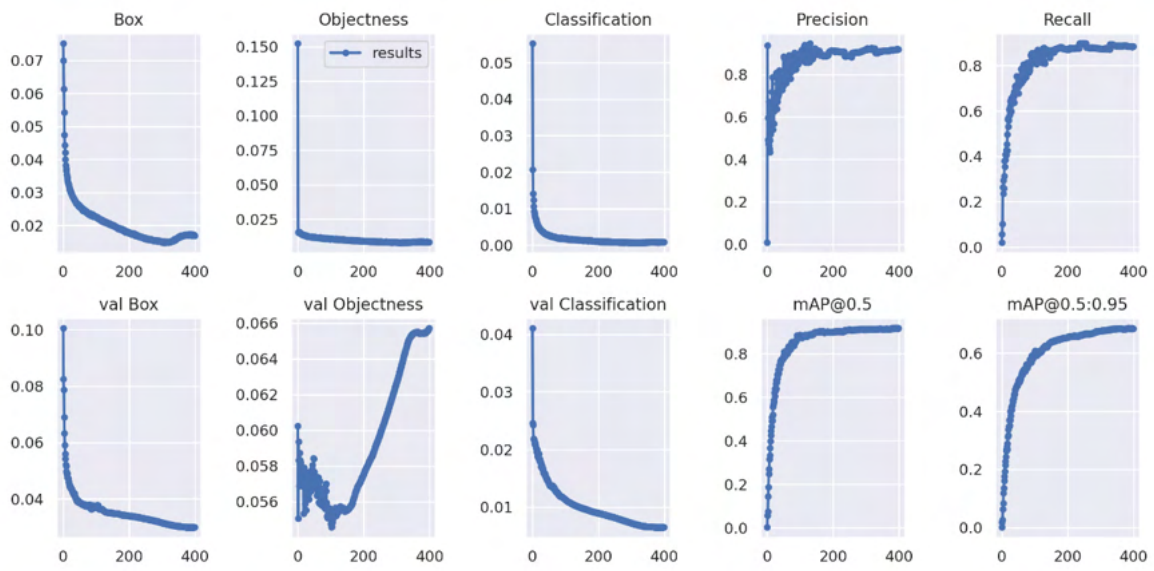


Figure 47

Epoch	Batch_size	Img_size	Precision	Recall	mAP@.5	mAP@.5:.95	Weight
+100	4	640	0.9186	0.884	0.9168	0.6856	Best_YOLOv7

Table 26

4.2.4 WAYMO dataset results without finetuning and layer freezing

Figure 48

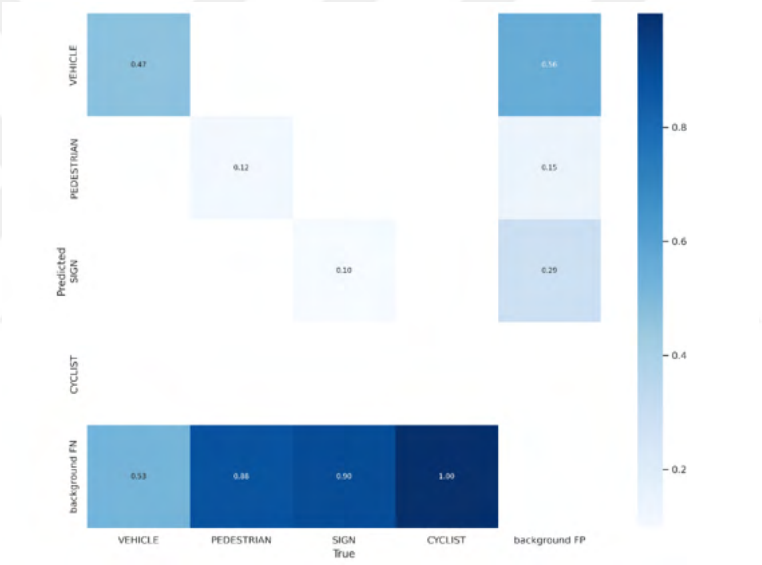


Figure 49

Table 27

Epoch	Batch_size	Img_size	Precision	Recall	mAP@.5	mAP@.5:.95	Weight
100	4	640	0.298	0.2406	0.2054	0.08621	YOLOv7

4.2.5 WAYMO dataset results with finetuning and first 50 layer freezing

Figure 50

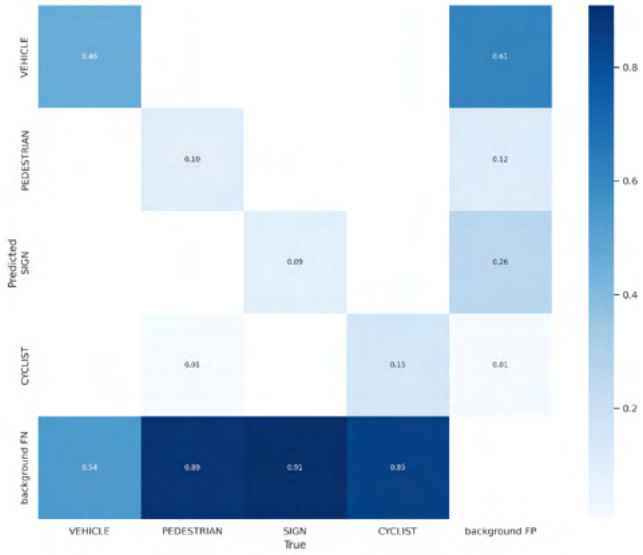


Figure 51

Table 28

Epoch	Batch_size	Img_size	Precision	Recall	mAP@.5	mAP@.5:.95	Weight
+100	4	640	0.5733	0.2494	0.2686	0.1179	Best_YOLOv7

4.2.6 WAYMO dataset results with finetuning and first 101 layer freezing

Figure 52

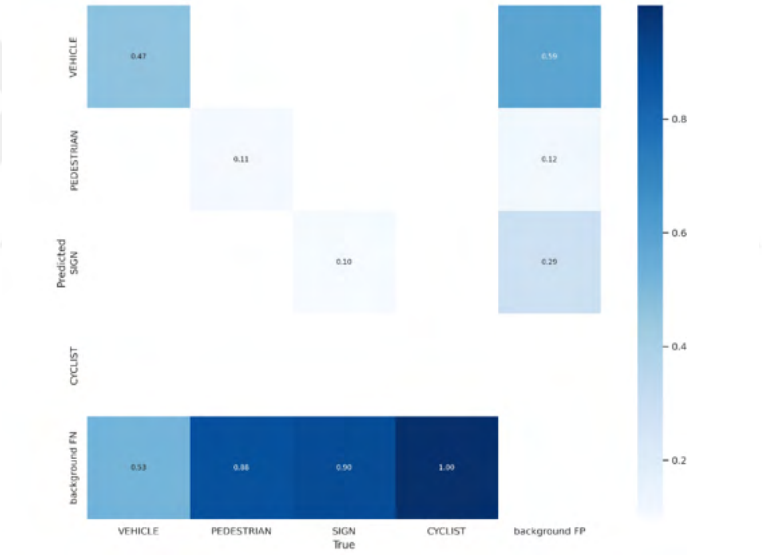


Figure 53

Epoch	Batch_size	Img_size	Precision	Recall	mAP@.5	mAP@.5:.95	Weight
+100	4	640	0.3058	0.2823	0.2199	0.08912	Best_YOLOv7

Table 29

4.2.7 Comparison of YOLOv7 training result

- First of all, compared to Scalled-YOLOv4, we got similar results because of the lack of performance of WAYMO dataset. Every graph reflects really bad results. So, we can add all result graphs.
- When we looked at the training result with KITTI dataset, there is no doubt that these result are really acceptable. For every type of metric, we got desirable result. Thanks to the fine tunning and layer freezing, we improve our accuaracy. We already discuss the reason why these technics help us to get better result. Moreover, when we looked at the results of loss values, these values are really small and we were desired this situation because if we have less loss then our training and validation accuaracy will increase. When we looked at F1 score, which is directly related to the recall and precision, as average, classes got 0.90 accuaracy at the threshold 0.457, which is acceptable result. Also, when we looked at the AP50 table, every classes have high accuaracy expect of person_sitting. Expect of person_sitting class, every class is higher than 0.80 value at AP50, which is perfect.

Table 30

Epoch	Batch_size	Img_size	Precision	Recall	mAP@.5	mAP@.5:.95	Weight
+100	4	640	0.9073	**0.8949**	**0.9174**	0.6852	Best_YOLOv7

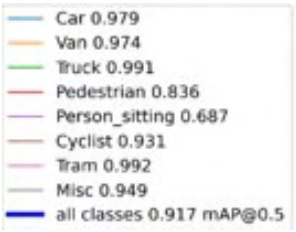


Figure 54

4.3 Simulation Results of Best trained algorithm

This is a result of YOLOv7 with KITTI dataset with finetuning and first 50 layer freezing. We got these results;

As we mentioned before, recall value is really important for us because if our FN value is high, our

recall value will be high. For example, if FN value is high, objects that should be classified as pedestrians are not classified as pedestrians. Therefore, in autonomous cars industry, this is unacceptable situation. Because of this reason and because of the highest mAP50, we selected this trained weight. We can see the simulation results below.

Figure 55

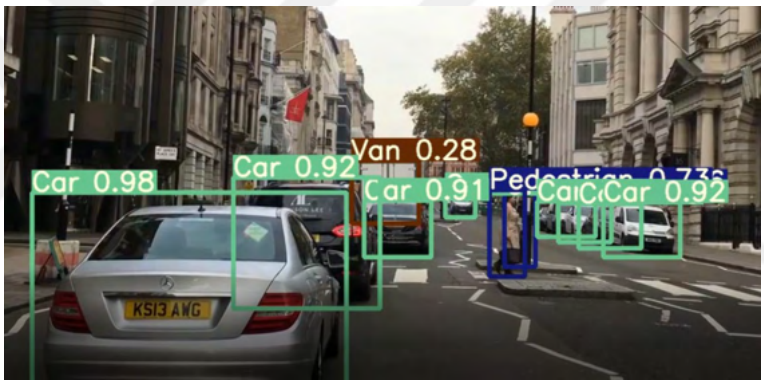


Figure 56

Figure 57

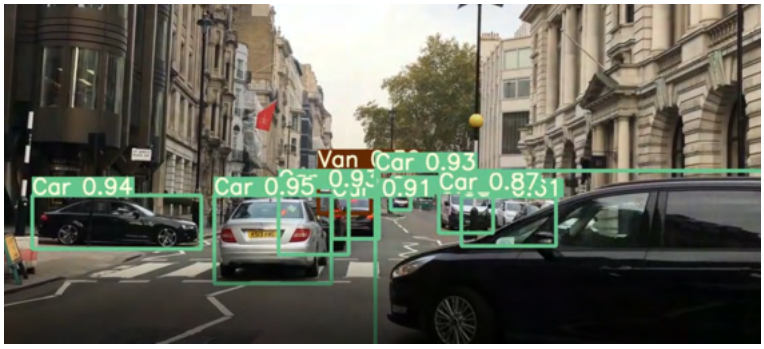


Figure 58

5 CONCLUSION

As can be understood from this report, the project aims to use CSP-sized version of scaled YOLOv4 and YOLOv7 versions and then with KITTI and WAYMO datasets, we trained these two algorithms. Also, we used fine tuning and layer freezing techniques to get better accuracy. After that we compare the results and then we select the best configuration. With these architectures, we will do that objects which may pose a problem for vehicles, pedestrians and autonomous vehicles on roads will be recognized, classified and tracked with "object detection, classification and tracking".

5.1 Future Work

After the completion of the object recognition, classification and tracking studies for autonomous vehicles, these studies on the autonomous vehicle in the CAV laboratory will be implemented. In addition, experiments and research will be carried out on whether this study can be done with the ZED2 camera for 3D object recognition, which is another very important issue for autonomous vehicles. Depending on the situation, it is planned to focus on this issue later.

REFERENCES

[1] A. Bochkovskiy, “Scaled YOLO v4 is the best neural network for object detection on MS COCO dataset,” Medium, 07-Dec-2020. [Online]. Available: https://alexeyab84.medium.com/scaled-yolo-v4-is-the-best-neural-network-for-object-detection-on-ms-coco-dataset-39dfa22fa982. [Accessed: 05-May-2022].

[2] “Lucas Nülle - CAV 2 automated guided vehicle system with collaborative robot arm,” Lucas-nuelle.us. [Online]. Available: https://www.lucas-nuelle.us/2768/apg/14295/CAV-2-automated-guided-vehicle-system-with-collaborative-robot-arm.htm. [Accessed: 05-May-2022].

[3] N. Cibik, “Training autonomous vehicles using augmented random search in Carla,” Towards Data Science, 27-Dec-2020. [Online]. Available: https://towardsdatascience.com/training-autonomous-vehicles-using-augmented-random-search-in-carla-19fcbe62b697. [Accessed: 05-May-2022].

[4] R. Gandhi, “R-CNN, fast R-CNN, faster R-CNN, YOLO — object detection algorithms,” Towards Data Science, 09-Jul-2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Accessed: 05-May-2022].

[5] “Risk priority number (RPN),” Leansixsigmadefinition.com. [Online]. Available: https://www.leansixsigmadefinition.com/glossary/risk-priority-number-rpn/. [Accessed: 05-May-2022].

[6] Open dataset – (no date) Waymo. Available at: https://waymo.com/open/ (Accessed: September 6, 2022).

[7] “The KITTI Vision Benchmark Suite,” Cvlibs.net. [Online]. Available: http://www.cvlibs.net/datasets/kitti/. [Accessed: 05-May-2022].

[8] K. Sambasivarao, “Non-maximum suppression (NMS),” Towards Data Science, 01-Oct-2019. [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c. [Accessed: 05-May-2022].

[9] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression,” in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 658–666.

[10] Keras Team, “VGG16 and VGG19,” Keras.io. [Online]. Available: https://keras.io/api/applications/vgg/. [Accessed: 05-May-2022].

[11] “No title,” Currency.com. [Online]. Available: https://currency.com/how-to-read-and-use-exponential-moving-average-ema-
indica-
tor?utm_medium=cpc&utm_source=googleads_pmax&utm_campaign=TR_PFM_FTD_EN&utm_

con-
tent=&campaignid=16609958636&adgroupid=&network=x&keyword=&matchtype=&creative=&adposi-
tion=&placement=&device=c&devicemodel=&extension=&loc_physical=1012782&gclid=CjwKCAjw682TBhATEiwA9crl38RySkNOIUdPVu3_1_giNOiJJSzynPowwRPMThCNc9CfdVevy9GiKxoC97YQAvD_BwE. [Accessed: 05-May-2022].

[12] Wang, C.-Y., Bochkovskiy, A. and Liao, H.-Y. M. (2020) "Scaled-YOLOv4: Scaling cross stage partial network," arXiv [cs.CV]. doi: 10.48550/ARXIV.2011.08036.

[13] Workingknowledge-csp.com. [Online]. Available: https://workingknowledge-csp.com/csp-model/. [Accessed: 05-May-2022].

[14] Wang, C.-Y., Bochkovskiy, A. and Liao, H.-Y. M. (2022) "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," arXiv [cs.CV]. doi: 10.48550/ARXIV.2207.02696.

[15] Tsang, S.-H. (2022) Brief review — SiLU: Sigmoid-weighted linear unit - sik-ho Tsang, Medium. Available at: https://sh-tsang.medium.com/review-silu-sigmoid-weighted-linear-unit-be4bc943624d (Accessed: September 6, 2022).