Department of Computer and Communication Systems Engineering

Faculty of Engineering

Universiti Putra Malaysia

ECC4306: MINI AI PROJECT PROPOSAL

TITLE: OBJECT DETECTION FOR ROBOTIC APPLICATION

Group Members: 1. Aina Afiqah binti Abdul Rahim (192158)

2. Muhammad Hazman bin Shahul Hameed (191991)

Lecturer : Prof. Madya Dr. Syamsiah bt. Mashohor

1.0 Background

Nowadays, there are many cases where individuals are unable to move their own bodies due to an injury or physical deterioration. Thus, more human capital needs to be invested in assisting their lives [1]. With various current technological advancements, the aim of assisting human population in doing routine jobs in everyday contexts can be achieved [2]. In this project, the focus is on the implementation of object recognition for robotic where the application involves assisting humans in picking up common objects in a room.

Therefore, the most important part in achieving that is by implementing the best algorithm with the least error as possible for a real environment. However, this is still a difficult objective as the implementation will be dealing with different kinds of objects and the environment surrounding the object itself. For instance, interaction with objects of multiple size, shape and degree of mobility can greatly affect the outcome. The algorithm chosen must be reliable enough to compute the input which is the visual data as well as reacting to changes in the environment in real time, whether it involves a simple or complex background [2].

For this project, a neural network framework which is known as Darknet will be used. As for the object detection and recognition, you only look once or better known as (YOLO), a real-time object detection system is chosen. It performs the detection by executing two different tasks which is identifying the object's position and classifying the class of the object. A dataset of 100 images per classes will be trained where 5 classes will be involved altogether. Finally, the performance of the trained dataset will be tested using YOLOv4.

2.0 Literature Review

2.1 Introduction

When humans look at an image, they instantly recognise the objects contained within, their location, and their interaction. The human visual system is quick and precise, which enables us to perform complex tasks such as driving with little conscious thought. Fast, accurate object detection algorithms would enable computers to drive cars with the use of specialised sensors, assistive devices to communicate real-time scene information to human users, and open the door to general-purpose or responsive robotic systems [3].

Recently, the number of research that focuses on the problems that involves object detection and recognition has tremendously increased. One of them being the difficulty of moving around, especially for elderly and cases where assist is constantly needed for them. Human Support Robot (HSR), assistive robots have been identified as a possible solution to provide the aid and enhance one's quality of life [2]. An advantage of using robots is that they can perform any tasks repetitively and do not get exhausted physically or emotionally as humans do [1].

2.2 Object Recognition/Detection System

Various areas have been explored and applied such as deep neural network (DNN), convolutional neural network (CNN), you only look once (YOLO), SSD, and RetinaNet. Each of them differs in the way that they process the image, which highly contributes to the performance and accuracy of the trained model. Basically, continuous improvements have been made to achieve a better result in terms of the robustness and effectiveness of the model by comparing to the existing techniques and each of the neural network uses different methods on object recognition.

2.3 You Only Look Once: Unified, Real-Time Object Detection

Fast, accurate object detection algorithms would enable computers to drive cars with the use of specialised sensors, assistive devices to communicate real-time scene information to human users and open the door to general-purpose or responsive robotic systems. As in the Figure 1 below, the system resizes the input image to 448 x 448 pixels and then runs a single convolutional neural network on it, with the output being the output model confidence threshold. YOLO works by combining separate components of object detection into a single neural network. The network predicts each bounding box using features from the entire image. Then, it predicts all bounding boxes for an image across all classes concurrently as shown in Figure 2. [7]

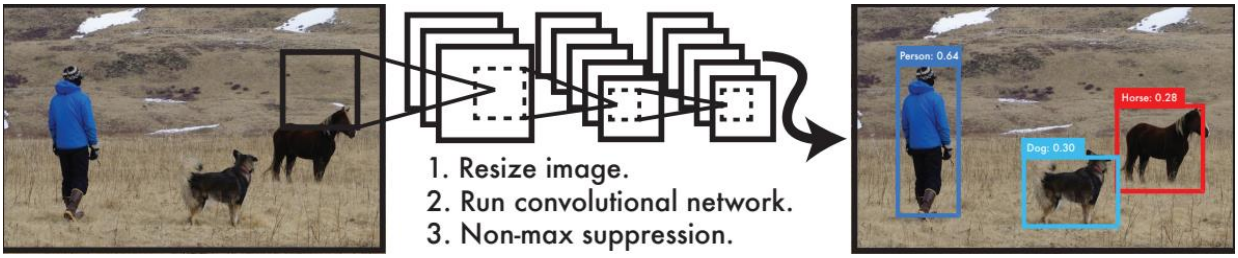

Figure 1: The YOLO Detection System

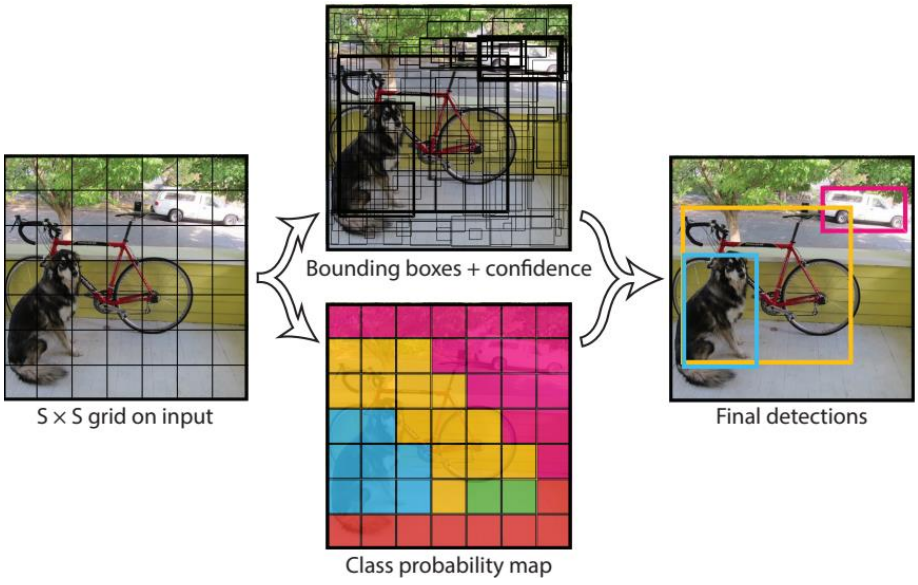

Figure 2: The Model

2.3.1 YOLOv4

YOLOv4 was an April 2020 publication of a real-time object detection model that achieved state-of-the-art performance on the COCO dataset. It accomplishes this by dividing the task of object detection into two parts, which are regression to estimate the object's position using bounding boxes and classification to establish the object's class. YOLOv4 is an evolution of the YOLOv3 model and an integration of many YOLO family's previous research efforts, as well as a number of new contributions specific to itself. It is implemented using the Darknet framework in the implementation of this mini project [11].

In comparison with the previous version, the AP (Average Precision) for YOLOv4 have improved by 10% and 12% for FPS (Frames Per Second. The architecture of YOLOv4 is composed of the backbone CSPDarknet53, a spatial pyramid pooling extra module, a PANet path-aggregation neck, and a YOLOv3 head [9]. Its overall architecture can be seen from Figure 1 below:

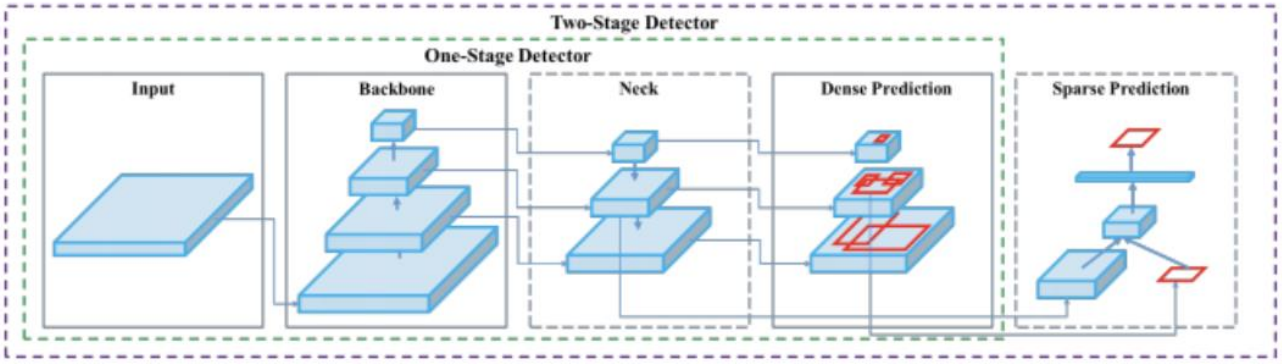

Figure 1: YOLOv4 Architecture [8]

YOLOv4 incorporates numerous latest features and integrates them to reach state-of-the-art results. The result from the Figure 2 below shows that the percentage for AP is 43.5 percent (65.7 percent AP50) for the MS COCO dataset at a real-time rate of 65 frames per second on the Tesla V100 [9]. Compared to other models that show advantages in terms of high performance or high interference speeds, YOLOv4 shows a well-balanced result which are incredibly high performance for a very high FPS.

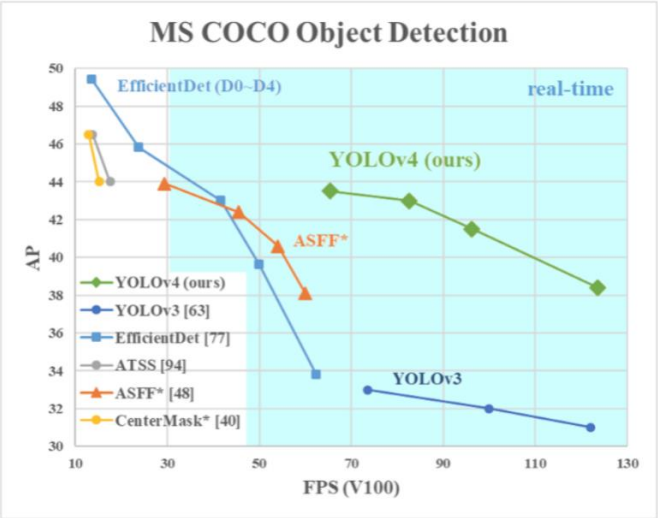

Figure 2: Yolov4 Results [9]

2.3.2 YOLOv4-tiny

YOLOv4-tiny is the compressed version of YOLOv4. Its simpler network structure and lesser parameters proposed from YOLOv4 makes it practical to be developed on mobile and embedded devices. It provides a significantly quicker training and detection. With two YOLO heads, as compared to three in YOLOv4, it was learned using 29 pre-trained convolutional layers, as opposed to 137 in YOLOv4 [9].

According to performance measures shown in Figure 3, YOLOv4-tiny is around 8X as quick as YOLOv4 at inference time and roughly 2/3 as performant on MS COCO dataset. On small custom detection tasks that are more tractable, you will see even less of a performance degradation. On custom detection tasks that are smaller and easier to control, the output will show lesser degradation in terms of performance [12].

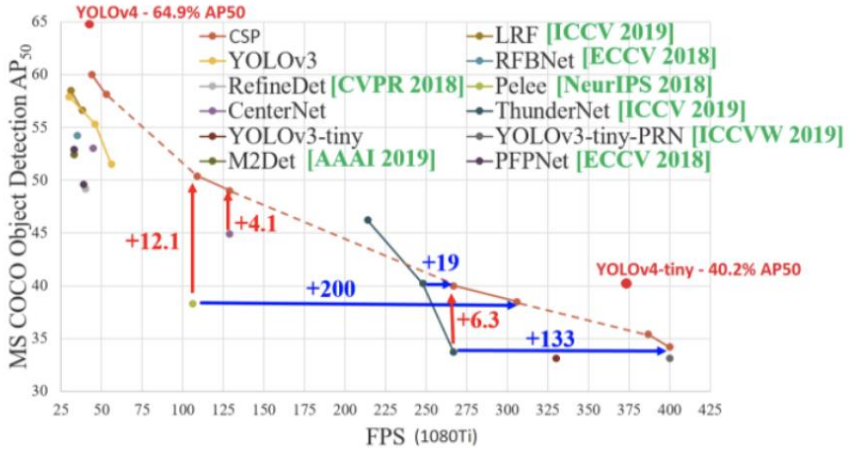

Figure 3: YOLOv4-tiny Performance Metrics [12]

As the project is focusing on implementing object detection for robotic application which will be real-time, YOLOv4-tiny is the better option when compared with YOLOv4. When it comes to a real-time object detection environment, the consideration is not mainly on precision or accuracy but also on which that has a faster inference time [9].

2.4 Object Recognition on Robotic Application

Object recognition and robotics has been a very interesting topics to be discussed as it will helps a lot to solve certain robot limitation in recognizing objects. Human-assisted robotics has been one of the fastest growing fields of technology. Among the domain's powerful features are object vision and robot manipulation. Vision-based manipulation has a lot of applications in human-assisted robotics. The proposed system has been evaluated using everyday objects. To recognise them, the system must first be trained on a dataset containing numerous images of each object [8]. On paper [9], it proposes an assistive navigation system to assist visually impaired users in corridor environments by considering the corridor's connectivity (corridors connect room doors and stairs). The YOLO neural network is used to detect and identify common indoor landmarks such as toilets, exits, and staircases, and the system can provide voice feedback about objects in its line of sight throughout the movement.

2.5 Training and Results

To recognize objects, the system must first be trained on a dataset containing numerous images of each object. The trained output will be the model that can be used to evaluate the result (consist of accuracy and speed). The training of the objects is accomplished using the YOLO v3 custom object dataset. To train each object, it is necessary to consider multiple images of that object that have been labelled using the annotation tool. Each object is photographed differently depending on its orientation, size, colour, and so on. For this study, a variety of images were taken.

Object	Number of images
Bottle	310
tomato	295
Cup	300
dining table	280

Table 1: Object (classes) and number of images to train

The recognition of trained objects used in the setup is visualized in Figure 3. The classification accuracy of the objects used in the experiment is represented in Figure 4. The accuracy is absolute for the cup, but 56 percent for the dining table, and 77 percent and 72 percent for the bottle and tomato, respectively. However, classification accuracy improves as the number of test cases increases and is dependent on the features used to implement YOLO [8].

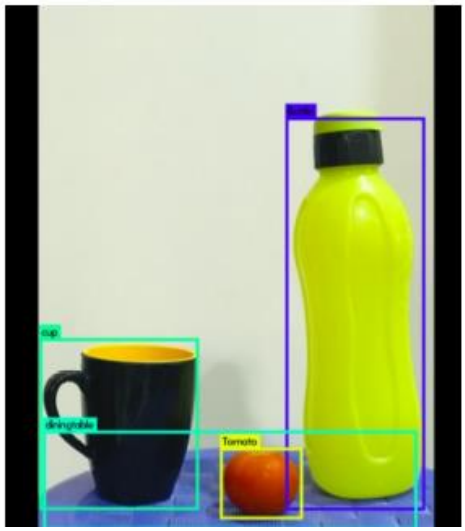

Figure 4: Object Recognition

```
   99 conv    128  1 x 1 / 1    76 x  76 x 384   ->    76 x  76 x 128
OPs
  100 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256
OPs
  101 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128
OPs
  102 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256
OPs
  103 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128
OPs
  104 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256
OPs
  105 conv    255  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 255
OPs
  106 yolo
Loading weights from yolov3.weights...Done!
data/huh.jpg: Predicted in 22.928922 seconds.
Bottle: 77%
diningtable: 56%
Tomato: 72%
cup: 100%
```

Figure 5: Classification accuracy of the objects

In paper [9], YOLOv3 is used in parallel with the Darknet framework in ROS to detect common objects in a corridor environment (such as chairs, people, and potted plants, among others) using pre-trained weights provided by Joseph Redmon. It is possible to have a frame rate of up to 24 frames per second. Additionally, we collect several types of signs as in Figure 6 and train them with YOLO to enhance the user's perception of the environment.

(a) Exit Signs

(b) Toilet sign

(c) Stair sign

Figure 6: The signs that used for training by [9].

3.0 Problem Statement

- People with physical deterioration or disability faces trouble in performing ordinary task such as picking an object.
- Less resources in human assistant that can provide continuous aid when needed.

4.0 Objectives

- To perform object detection using YOLOv4-tiny for the application of human assistive robot.
- To train a dataset of 160 images per class to be able to detect 10 objects in a room.
- To compare, verify and test the performance of the trained datasets.

5.0 Methodology

5.1 Introduction

This section will focus on the way of the project is being carried out from the execution stage until the submission stage. Figure below shows the block diagram of the project that will be show the step required to achieve the objective of this project.

<table>
<tr><td colspan="3">Object Recognition for Robotic application using Computer Vision</td></tr>
<tr><td>Framework</td><td>Dataset</td><td>Training and testing</td></tr>
<tr><td> 
</td><td> 
</td><td> 
</td></tr>
</table>

Table 2: Block diagram of the methodology

5.2 Installation and configuration

5.2.1 Framework

This project will be using Darknet Framework for neural network framework. Darknet is an open-source framework for neural network that is developed using CUDA and C. Darknet can be run using CPU or GPU power computational and were built initially for Linux or Mac computer OS.

5.2.2 Object Recognition/Detection System

For the Object Recognition/Detection system, this project will be using YOLO: Real Time Object Detection system, an object detection system that achieves a very good result comparing to other object detection systems. YOLO works by splitting task on object detection into two separate tasks. The first one is regression uses to identify object position via bounding boxes and another one is classification to determine object class. This project will be using YOLOv4-Tiny but we will be comparing with YOLOv4 to give a better view for the result.

5.2.3 Training Platform and Configuration

For the training platform, this project will be using Google Collab to set up the environment and dataset training. Google Collab is a Python Jupyter that can do training with GPU capabilities. Before installing the Darknet, GPU Driver, OpenCV and cuDNN need to be installed and configured if it is not installed yet. Then Darknet along with the convolutional neural network used in YOLOv4 will be downloaded and installed. Both of the model uses the same dataset but different pre-trained model and configuration. To get the training weight, YOLOv4 uses 137 layers pre-trained model while YOLOv4 uses 29 layers pre-trained model. The configuration of each of the model is as below:

Configuration	YOLOv4	YOLOv4-Tiny
Max batch	20000	20000
Burn-in	1000	1000
Batch	64	64
Subdivision	16	16
W x H	608 x 608	418 x 418
Learning rate	0.0001	0.000261

The iteration is based on max batch and it is based on the formula which is 2000 x No. of class. 16 Subdivision is to achieve the best result of the training without limiting the memory bandwidth on Google Collab.

5.3 Dataset

The dataset is fetched from Google Open Images V4. Google Open Images is images dataset contains 600+ classes including annotation label. The dataset that will be used in this project is Open Images V4 [1]. The dataset per classes will be at 100 images per classes for the first trial. Images will be downloaded using class function. The dataset obtained will be based on the class names and training dataset which will be the object as in the methodology list. The dataset will be fetched using OIDv4 Tool which can specify preferred object and the specification of the images which is the image is not occluded and is not from the inside view. The annotation of the dataset is then converted from standard annotation format to YOLO TXT format using Python Script. The images is then will be resized by YOLOv4 into the configured format to ensure the result is unaffected by the pixels.

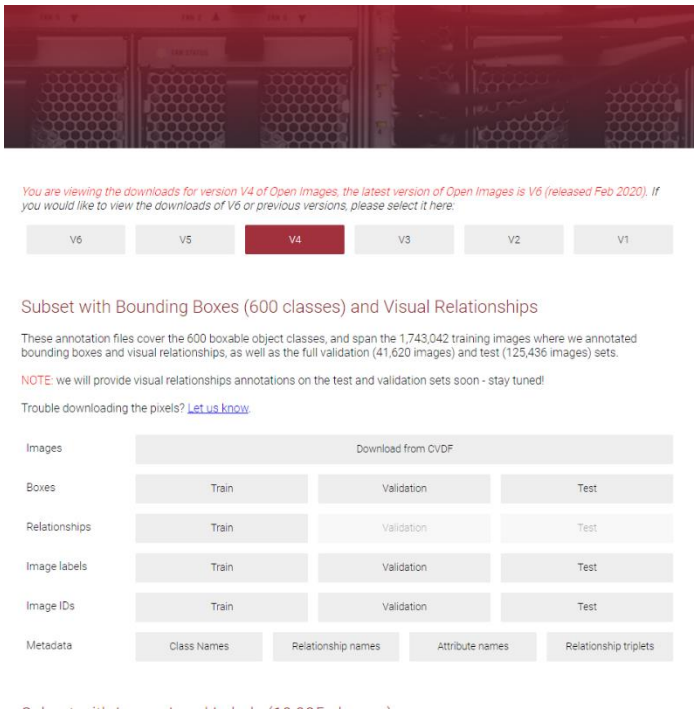

Figure 7: Open Images V4 specific subsets

Figure 8: OIDv4 and annotation converter tool.

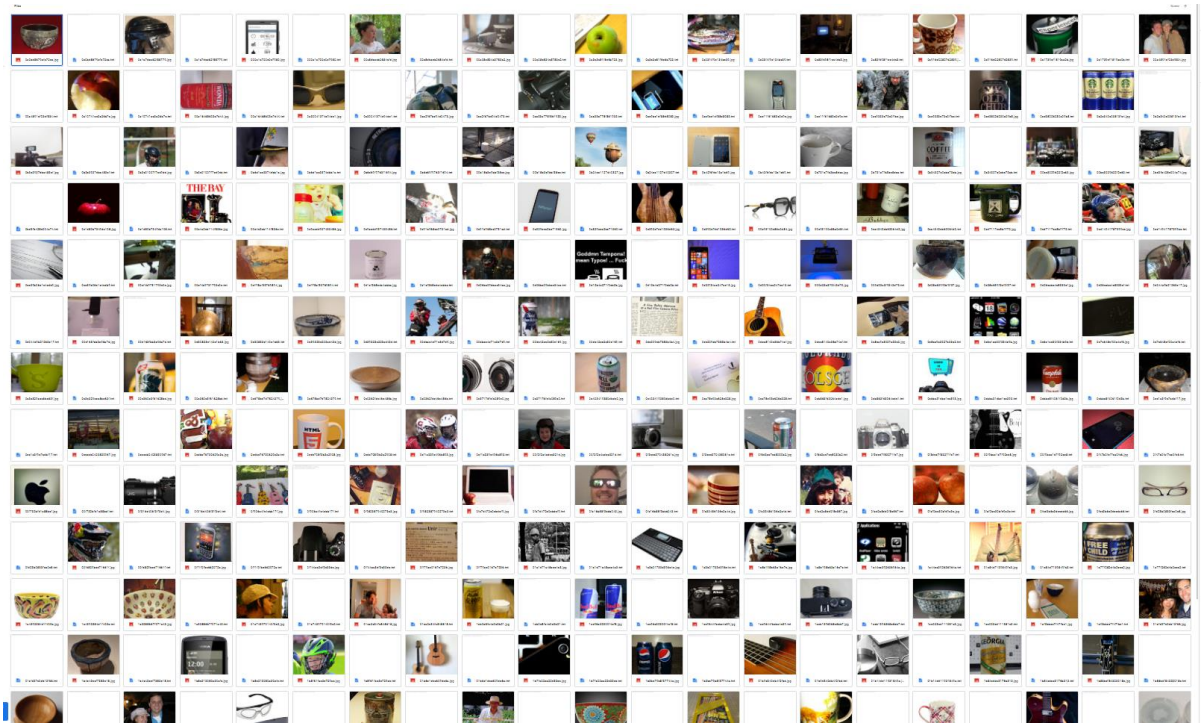

Figure 9: Images and Label for the dataset

5.4 Training and Testing

The data is then will be trained using Darknet training environment. The mean average precision (mAP) will be taken and recorded. Then the weight that has the highest mAP will be used to test the object detection using YOLOv4-tiny. There will be 10 objects that will be trained for the Object Recognition System to recognize. The split ratio for train and validation is 80:20. YOLOv4 calculate mAP and do validation starting on the 1000 iteration and on every 100 iteration until it is finished. The list of the object will be as the Table 3 below:

Object	Validation Images	Train Images
Apple	40	160
Guitar	40	160
Mobile Phone	40	160
Hat	40	160
Watch	40	160
Bowl	40	160
Tin can	40	160
Helmet	40	160
Camera	40	160
Coffee	40	160

Table 3: List of object and images that will be used to train.

Due to limited resources available on Google Collab, the result is obtained by the maximum time training time allowed by Google Collab and the training is ended at 14000 iteration on YOLOv4-Tiny and 3500 iteration on YOLOv4. The time taken to reach the number of iteration is 5 hours for both of the model.

6.0 Result and Discussion

For the result, we train both YOLOv4 and YOLOv4-Tiny model to compare the difference between two framework. Both model produce different result and YOLOv4 produces a better result as expected.

6.1 The result (Average Precision) obtained by each of the model were:

YOLOv4

Object	Average precision (%)	TP	FP
Apple	80.72	41	21
Guitar	94.19	43	8
Mobile Phone	90.76	45	5
Hat	82.98	29	3
Watch	82.66	34	8
Bowl	89.87	38	77
Tin can	90.65	44	9
Helmet	73.74	48	15
Camera	90.37	41	12
Coffee	89.91	35	4

YOLOv4-Tiny

Object	Average precision (%)	TP	FP
Apple	66.60	38	20
Guitar	70.18	32	15
Mobile Phone	78.43	39	18
Hat	57.67	20	5
Watch	59.99	24	10
Bowl	63.67	28	15
Tin can	72.35	37	18
Helmet	50.57	21	13
Camera	65.16	35	20
Coffee	73.59	27	13

From the Average Precision results, it can be seen that YOLOv4 has better Average Precision as it is trained with smaller learning rate, bigger image pixel and is trained using 137 layers pre-trained model comparing to YOLOv4-Tiny. Even on 3500~ iteration, YOLOv4 manages to recognize the object and predict them accurately comparing to YOLO-Tiny at 14000~ iteration.

6.2 Summary of mAP and classification result

YOLOv4

Confident threshold	mAP	precision	recall	F1-Score	IoU (%)
0.25		0.81	0.82	0.81	64.82
0.5	0.86				

YOLOv4-Tiny

Confident threshold	mAP	precision	recall	F1-Score	IoU (%)
0.25		0.67	0.63	0.65	51.53
0.5	0.65				

To summaries, the difference on mAP at confident threshold of YOLOv4 and YOLOv4-Tiny is almost at 25%. Comparing to the result obtained with the other project that comparing YOLOv4 and YOLOv4-Tiny, the difference is within the range which is 2/3 accuracy which has been stated on [12]. Based on the following mAP on iteration chart below, it can be seen that after 6000 iteration, the training starting to show overfitting and the accuracy becomes fluctuating.

6.3 Mean Average Precision (mAP) throughout the iteration

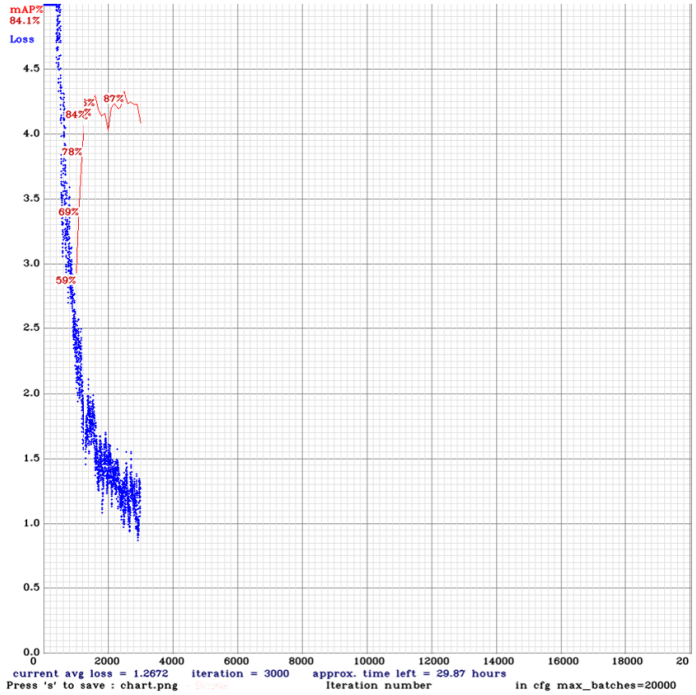

Figure 9: mAP for YOLOv4

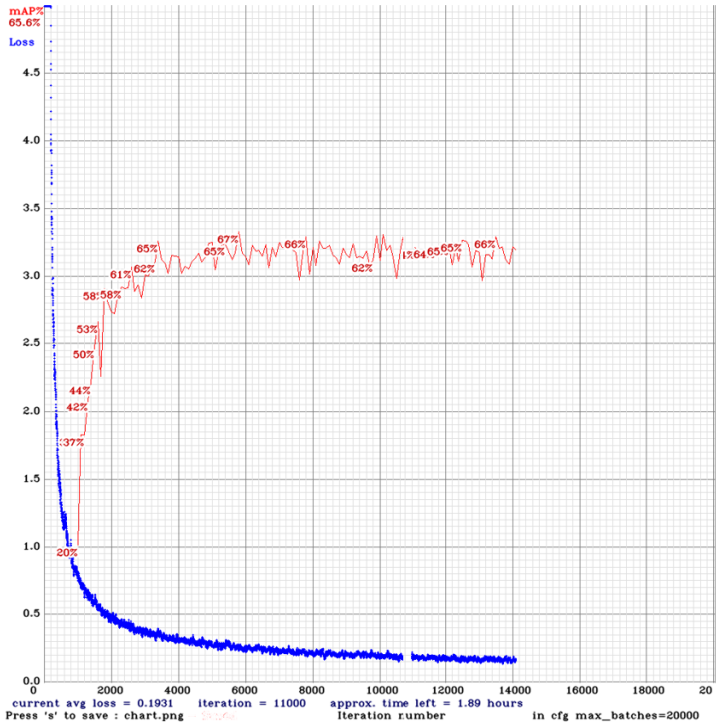

Figure 10: mAP for YOLOv4-Tiny

6.4 Further Discussion

The result is affected by many factor which is the number of images per class in the dataset, the quality of the images and the image itself. Based on the observed images in the dataset, all of the images is clear, visible and under clear lighting. For certain classes such as Hat and Helmet, most of the images include human head which might confuse the model training as the dataset is not wide enough to ensure the model knows which class belongs to the images.

On the darknet framework, YOLOv4/YOLOv4-Tiny uses number of batches to calculate its iteration and it does the validation using the validation images on after every 100 iteration carried out. As YOLO uses single stage detection, it will produce a less accurate results comparing to other framework that uses two stage detection.

On the performance, YOLOv4 is proven to produce a fast result with the drawback of accuracy. Unfortunately, the test phase is unable to be carried out due to hardware limitation.

7.0 Conclusion

YOLOv4 and YOLOv4-Tiny has a different layer architecture and the result is 25% difference. The accuracy can be increased with more wide images for the dataset, higher iteration, and smaller learning rate. The result cannot conclude which model has a better result as both of them serves different needs even though both of the model uses to detect same number object.

8.0 References

[1] J. Y. Yang, U. K. Chen, K. C. Chang, and Y. J. Chen, “A novel robotic grasp detection technique by integrating YOLO and grasp detection deep neural networks,” in International Conference on Advanced Robotics and Intelligent Systems, ARIS, Aug. 2020, vol. 2020-August, doi: 10.1109/ARIS50834.2020.9205791.

[2] E. Martinez-Martin and A. P. Del Pobil, “Object detection and recognition for assistive robots: Experimentation and implementation,” IEEE Robot. Autom. Mag., vol. 24, no. 3, pp. 123–138, Sep. 2017, doi: 10.1109/MRA.2016.2615329.

[3] "Open Images," Open Images, [Online]. Available: https://storage.googleapis.com/openimages/web/index.html. [Accessed 10 June 2021].

[4] Alphabet Inc., "Google Collab," Google, [Online]. Available: https://colab.research.google.com. [Accessed 10 06 2021].

[5] Jupyter Notebook, "Jupyter," [Online]. Available: https://jupyter.org/. [Accessed 10 June 2021].

[6] 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)*, Las Vegas, NV, 2016.

[7] R. K. Megalingam, R. V. Rohith Raj, T. Akhil, A. Masett, G. N. Chowdary and V. S. Naick, "Integration of Vision based Robot Manipulation using ROS for Assistive Applications," in *Proceedings of the 2nd International Conference on Inventive Research in Computing Applications, ICIRCA 2020*, 2020.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection.” Accessed: Jul. 03, 2021. [Online]. Available: https://github.com/AlexeyAB/darknet.

[9] “YOLOv4 vs YOLOv4-tiny. Training custom YOLO detectors for Mask… | by Techzizou | Analytics Vidhya | Medium.” https://medium.com/analytics-vidhya/yolov4-vs-yolov4-tiny-97932b6ec8ec (accessed Jul. 03, 2021).

[10] “YOLO: Real-Time Object Detection.” https://pjreddie.com/darknet/yolo/ (accessed Jul. 03, 2021).

[11] “YOLOv4 Darknet Object Detection Model.” https://models.roboflow.com/object-detection/yolov4 (accessed Jul. 03, 2021).

[12] “Train YOLOv4-tiny on Custom Data - Lightning Fast Object Detection.” https://blog.roboflow.com/train-yolov4-tiny-on-custom-data-lighting-fast-detection/ (accessed Jul. 03, 2021).

[13] Techzizou, “YOLOv4 VS YOLOv4-tiny,” Medium, 30-Jun-2021. [Online]. Available: https://medium.com/analytics-vidhya/yolov4-vs-yolov4-tiny-97932b6ec8ec. [Accessed: 03-Jul-2021].