International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Object Detection for Autonomous Vehicles with Sensor-based Technology Using YOLO

Nurin Miza Afiqah Andrie Dazlee[1], Syamimi Abdul Khalil[2], Shuzlina Abdul-Rahman[3*], Sofianita Mutalib[4]

Submitted: 24/08/2021 Accepted : 10/02/2022

Abstract: The year 2020 has been a tough year with the global pandemic situation, and the utmost priority is to live in a clean, green, and safe environment. One of the areas that the governments are emphasizing for the readiness of our ecosystem is autonomous and contactless environments in adapting to the new norm. Thus, Autonomous Vehicle (AV) is a promising technology to bring forward. One of the critical aspects of Autonomous Navigation is object detection. Most AV use multiple sensors to detect objects, such as cameras, radar and Light Detection and Ranging sensor (LiDAR). Nowadays, the LiDAR sensor is widely implemented due to the ability to detect objects in the form of pulsed lasers, benefiting in low-light object detection. However, even with advanced technology, poor programming can affect the performance of object detection system. Thus, the study explores the state-of-the-art of You Only Look Once (YOLO) algorithms namely Tiny-YOLO and Complex-YOLO for object detection on KITTI dataset. Their performances were compared based on accuracy, precision, and recall metrics. The results showed that the Complex-YOLO has better performance as the mean average precision is higher than the Tiny-YOLO model when tested with equal parameters.

Keywords: *Autonomous Vehicle, KITTI, LiDAR, Object Detection, Sensor, YOLO.*

1. Introduction

To date, Autonomous Vehicles (AV) is a fast-growing technology and has sought the attention of many global vehicle companies [1]. This technology enables the vehicle to be on autopilot and navigate itself with little or zero human input. Nevertheless, there are many challenges in implementing the technology, which is why there are not many AVs roaming freely on the roads, especially in Malaysia. On March 18, 2018, it was reported that a test vehicle from Uber Technologies, Inc., operating with a self-driving system, killed a pedestrian in Tempe, Arizona [2]. Uber claims that the car was working ideally. The probable cause for the crash is poor programming as they did not equip an alert system for possible collision, instead of depending solely on the attention of the safety driver [2]. Therefore, this proves that there are problems with AVs, and it is vital to improvise every aspect of Autonomous Navigation.

One of the critical aspects of Autonomous Navigation is Object Detection. Effective detection is essential as they need to detect road elements and pedestrians before they can understand and respond to their surroundings. Some of the challenges of object detection in AVs are detecting objects in low-light conditions and slick surfaces. Other challenges would be poor resolution of data obtained by specific devices. For example, AVs use radar for object detection as it can handle low-light and adverse weather conditions. However, the problem with radar is that in some cases, they cannot distinguish pedestrians, especially children [3].

These days auto manufacturers have implemented Light Detection and Ranging (LiDAR) sensor for AVs due to the advantage of detecting objects in low light conditions as camera-based systems offer dense light projection but lack distance information [4]. However, even with the best performing sensors, most systems associated with AVs lack accuracy for complete self-drive because of the limitation of algorithms. Highly claimed for its performance, You Only Look Once (YOLO) algorithm, is seen as promising [5], but there is a need to prove these claims. The limitation for AVs to detect objects precisely will probably endanger the safety of both vehicle occupants and surrounding pedestrians. Thus, this study aims to interpret 3D image data from the sensor-based technology to train LiDAR and image data for object detection and to compare the performance of YOLO models by varying the epochs and the amount of data.

This paper is organized as follows: Section 2 presents information about sensor-based technology and object detection in past researches and ends with information about YOLO models. Section 3 describes the study's methodology, while Section 4 discusses the results and discussion of this research. Lastly, section 5 concludes the research.

2. Literature Review

This section provides the review of past research, discussing the advantages and disadvantages of both camera and LiDAR sensors, the role of object detection for the implementation of Autonomous Vehicles and ends by comparing Complex and Tiny-YOLO algorithms.

[1] *Faculty of Computer and Mathematical Sciences, UiTM Shah Alam, Selangor, Malaysia. ORCID ID : 0000-0002-2180-4804*
[2] *Faculty of Computer and Mathematical Sciences, UiTM Shah Alam, Selangor, Malaysia. ORCID ID : 0000-0002-9145-4280*
[3] *Faculty of Computer and Mathematical Sciences, UiTM Shah Alam, Selangor, Malaysia. ORCID ID : 0000-0002-5498-9606*
[4] *Faculty of Computer and Mathematical Sciences, UiTM Shah Alam, Selangor, Malaysia. ORCID ID : 0000-0001-8384-3131*
* *Corresponding Author Email: shuzlina@fskm.uitm.edu.my*

2.1. Sensor-based Technology

Autonomous Vehicles or AV use a variety of sensors to detect their surroundings. Cameras are widely used for AV object detection due to their low price and low cost in processing the data [6]. One advantage of using cameras is that it utilizes the same method of human vision. They perceive the situation by interpreting colours, reading street signs as humans do [7]. However, cameras would also imply problems in object detection as they function poorly in low-light conditions [3]. Another paper also stated the camera's limitations in lighting condition change because it has the same problems as human vision; the data would not be as in-depth as LiDAR data [6].

Therefore, LiDAR has been a promising sensor-based technology in auto manufacturing to overcome this issue. LiDAR uses lasers to calculate the time taken for each pulse to bounce back from the object to the sensor, producing a 3D map with a detailed shape. Using the time obtained, the distance can be calculated, and mapping is performed [8]. The compact measurement size of LiDAR enables online data transfer and processing; thus, it can record accurate and high framerate point cloud sequences from large environments [9].

Although there are many advantages to implementing LiDAR technology, there are also some downsides. It is said that LiDAR lacks in coverage and range compared to cameras. LiDAR are also known to have some issues in rainy conditions because it can cause glare and reflections [3][10]. Another disadvantage is that LiDAR is expensive compared to other sensors such as cameras and radars. However, these days the cost for LiDAR has shown to be cheaper than before, and the efficiency of the technology makes it worth investing on.

2.2. Object Detection for Autonomous Vehicles

Object detection is essential for AVs because, to drive safely, AVs need to detect their surroundings efficiently. At present, object detection is paired mainly with deep learning models to perform feature extractions, classifications, and localization (Jiao et al., 2019). In 2019, research presented a deep learning-based method of 3D person detection using camera images and LiDAR point clouds. It is stated that LiDAR is proposed as camera-based systems tend to impose challenges caused by weather changes and illumination. The proposed architecture is inspired by Aggregate View Object Detection (AVOD). The results stated that on moderate difficulty, using KITTI 3D as the benchmark produced an average precision of 46.06% [4].

Another interesting research that also implemented LiDAR to solve the camera's low-light problem uses deep learning algorithms. The algorithms used were You Only Look Once v2 and Single Shot Detection (SSD). It is shown that YOLOv2 is better at detecting objects in a single frame than SSD [1]. However, their study lacks in making the model being able to process video data.

2.2.1. Datasets Used

An abundant of research have collected their own data when studying LiDAR data which serves as an advantage since LiDAR data is quite hard to find online. It is undeniable that there are some LiDAR datasets are available online, although not all are accessible for free.

One of the most popular used datasets for 3D object detection specifically for autonomous cars is the KITTI dataset. The KITTI dataset created with a 64-channel LiDAR, provides large scale data in front camera view and LiDAR point cloud data. It contains three classes which are Class Car, Pedestrian and Cyclist, however, it has been observed that there are more cars in the data compared to pedestrians and cyclists. This may lead to bias an inaccuracy in training of data.

Other datasets are NuScenes and the H3D dataset by Honda, which are larger compared to KITTI. NuScenes contains one million bounding boxes while H3D contains 1.4 million, which is about 5 times than KITTI. Like KITTI, H3D was created with 64-channel LiDAR while NuScenes are created with 32-channel LiDAR. The more the number of channels, the denser and more complete 3D maps produced. Thus, this means that it will likely be more difficult to detect far-away objects as the beam divergence is larger [11].

2.2.2. YOLO Algorithms

Object detection algorithms are categorised as either classification-based (2-stage detectors) or regression-based (1-stage detectors). Classification-based algorithms detects objects in two stages; the first stage marks a set of regions if interest where the probability of objects is high while the second stage finds objects in the marked area by implementing Convolution Neural Networks (CNN). In contrast, single stage detectors or regression-based algorithms like You Only Look Once (YOLO) detects objects in one go. They do not select region of interest, instead, they predict classes by bounding boxes, which makes detection faster. Therefore, YOLO has been a popular choice for fast and accurate object detection.

First founded by Joseph Redmon et al. in 2015 [12], YOLO is a regression-based algorithm that is proven effective for end-to-end training and real-time detection, while maintaining high average precision. Ever since it launched, it is cited more than 16 thousand times and researchers have been implementing this algorithm in multiple use cases, even in detecting objects for Autonomous Vehicles and in intelligent video analytics.

YOLO divides images into grids, which contains equal dimensional region of S x S. The grid cell where the centre of an object is located, will be the cell assigned for detection. In each grid cell, B bounding boxes, the coordinate (x,y), height (h) and width (w) of the centre point and confidence scores (cs) are the predictive elements. Also, each cell has conditional class probabilities (C) used to predict to which class the object detected belongs to. Confidence scores (cs) reflect the model's confidence whether the box contains an object. If there are no objects in a particular cell, then the confidence score should be zero. Or else, the confidence score would be equals to the intersection over union (IOU) between the predicted box and the ground truth. The IOU is the calculation of area overlapped divided by the area of union, which provides a good estimate on how close the bounding box is to the original prediction. Figure 1 shows the basic structure of YOLOv3.

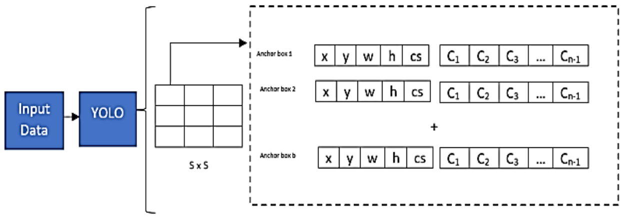


Fig. 1. Basic Structure of YOLOv3 algorithm [1]

Complex-YOLO is a state-of-the-art 3D object detection that works best in delivering point clouds of any surrounding environment in real-time. The algorithm is a 3D version of YOLOv2 that calculates the distance for any enclosed objects instantaneously. As point cloud processing is immensely significant for autonomous driving, Complex-YOLO has come

with an algorithm that gives approximate values of position and directions of different objects precisely in 3D [13].
Complex-YOLO has been proposed and tested to meet the needs of autonomous vehicles adequately. Its feasibility in object detection holds the potential solutions for problems that might occur in the said field as it can detect multiple objects just by a single inference [14]. On top of that, the multi-stage detector also complements any lower accuracy detector as it consists of convolution layers composed of a deep network. For point cloud-based 3D object detection, its complexity predominantly shows the irregularity of the point clouds from LiDAR sensors. As LiDAR is one of the common 3D sensors in autonomous driving, it is crucial to have a precise solution [15]. Another property of Complex-YOLO is its 3D detection methods that support a 2D detection framework in projecting point clouds from different types of views.
With a backbone network like the Darknet-53 network, Complex-YOLO works efficiently to present accurate detections but known to acquire complexity which requires exceedingly high computational power for hardware. For that reason, detection speed is highly affected. Thus, another simplified algorithm was introduced to overcome the issue. Tiny-YOLOv3 is technically a simplified algorithm of YOLOv3 that enhances the speed of object detection while maintaining its accuracy [16][17]. It is a real-time detector designed for low performance of data processing in a device. With a backbone network consisting of seven convolutional layers and six pooling layers, it improves the detection speed. In cases where it might lose detection accuracy, it is undeniably a good substitute for a costly algorithm [18]. Experiments like forming a three-scale detection from a two-scale and replacing traditional convolution with a deep separable convolution are still ongoing to counter Tiny-YOLOv3 limitations [19][20].

3. Methodology

This section describes the methodology of the study, which comprises two main phases: Data Collection & Preparation and Model Architecture & Training, which is done in Python language using Google Colaboratory with GPU.

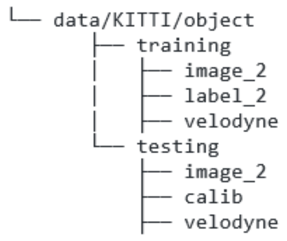
3.1. Data Collection & Preparation

For this study, the KITTI dataset is used. KITTI dataset is captured around Karlsruhe, within rural areas and highways with up to 15 cars and 30 pedestrians are visible per image [21]. Using an eight-core i7 computer equipped with a RAID system running Ubuntu Linux and a real-time database, Geiger et al. recorded the data using multiple sensors attached to a vehicle. The sensors used include one Inertial Navigation System (GPS/IMU), one Laserscanner, four Varifocal lenses, two Grayscale cameras and two Colour cameras with 1.4 megapixels. Firstly, KITTI data was transferred to Google Drive to be mounted in Google Colaboratory. To prepare the data, the directory structure was set as shown in Figure 2.

3.2. Model Architecture and Training

Point cloud data is converted to BEV image using python NumPy functions. BEV image is passed through the YOLOv3 network. Then, YOLO output is decoded, and results are drawn [22][13]. The model for both Complex-YOLO and Tiny-YOLO is made from many types of layers with its functionalities. In the configuration file of both models, there are layers described as convolutional, shortcut, route, upsample and YOLO. The architecture of YOLO models are shown in Table 1 and 2.

Fig. 2. Data Directory



```
└── data/KITTI/object
        ├── training
        │   ├── image_2
        │   ├── label_2
        │   ├── velodyne
        └── testing
            ├── image_2
            ├── calib
            ├── velodyne
```

Table 1. Complex-YOLO Architecture

Layer		Filters	Size	
YOLO Layer 1	conv	32	3x3/1	x2
	conv	64	3x3/2	
	shortcut	from -3	activation=linear	
	conv	128	3x3/2	
	conv	64	1x1/1	
	conv	128	3x3/1	
	shortcut	from -3	activation=linear	
	conv	64	1x1/1	
	conv	128	3x3/1	
	shortcut	from -3	activation=linear	
	conv	256	3x3/2	
	conv	128	1x1/1	
	conv	256	3x3/1	
	shortcut	from -3	activation=linear	
	conv	128	1x1/1	x7
	conv	256	3x3/1	
	shortcut	from -3	activation=linear	
	conv	512	3x3/2	
	conv	256	1x1/1	
	conv	512	3x3/1	
	shortcut	from -3	activation=linear	
	conv	256	1x1/1	x7
	conv	512	3x3/1	
	shortcut	from -3	activation=linear	
	conv	1024	3x3/2	
	conv	512	1x1/1	
	conv	1024	3x3/1	
	shortcut	from -3	activation=linear	
	conv	512	1x1/1	x4
	conv	1024	3x3/1	
	shortcut	from -3	activation=linear	
	conv	512	1x1/1	x3
	conv	1024	3x3/1	
	conv	30	1x1/1	
YOLO Layer 2	conv	256	1x1/1	
	upsample		stride=2	
	route		layers= -1,61	
	conv	256	1x1/1	x3
	conv	512	3x3/1	
	conv	30	1x1/1	
YOLO Layer 3	route		layers= -4	
	conv	128	1x1/1	
	upsample		stride=2	
	route		layers= -1,36	
	conv	128	1x1/1	x3
	conv	256	3x3/1	
	conv	30	1x1/1	

The layers in Tiny-YOLO are quite similar to that of Complex-YOLO, but there is a type of layer called Max pool. Maximum pooling calculates the most significant value in each patch of each feature map. Then, the output is pooled so that it highlights the most present feature in the patch. The architecture of Tiny-YOLO model is shown in Table 2.

Table 2. Tiny-YOLO Architecture

Layer		Filter	Size
YOLO Layer 1	conv	16	3x3/1
	maxpool	2x2	
	conv	32	3x3/1
	maxpool	2x2	
	conv	64	3x3/1
	maxpool	2x2	
	conv	128	3x3/1
	maxpool	2x2	
	conv	256	3x3/1
	maxpool	2x2	
	conv	512	3x3/1
	maxpool	2x1	
	conv	1024	3x3/1
	conv	256	1x1/1
	conv	512	3x3/1
	conv	30	1x1/1
YOLO Layer 2	route	layers=-4	
	conv	128	1x1/1
	upsample	stride=2	
	route	layers= -1	
	conv	256	3x3/1
	conv	30	1x1/1

We trained and tested the data in three experiments to study the Complex-YOLO and Tiny-YOLO models' performance. We have set the parameters according to the limitation in Google Colaboratory, limiting the number of data used and runtime using GPU. The parameters for each experiment are shown in Table 3.

Table 3. Experimental Setup

Experiment	Constant Variable	Independent Variable	
A	Total Data: 1000	*Training Data size*	*Testing Data size*
		800	200
		500	500
		700	300
		600	400
B	Train/Test: 800/200 500/500	*Epochs*	
		25, 50	
C (Compared Complex-YOLO with Tiny-YOLO)	Epochs: 50	*Training Data size*	*Testing Data size*
		800	200
		500	500

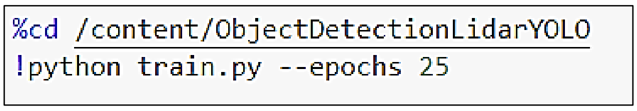
Based on the above parameters, this study evaluates the model's performance according to four metrics: class accuracy, precision, recall50 and recall75, as shown in Table 4.

Table 4: Performance Measures

Metric	Formula
Class Accuracy	100 * Mean of Class Mask [Object Mask]
Precision	Sum of (iou[a]50 * Detected Mask) / (Sum of Confidence Score + 1e-16)
Recall50	Sum of (iou50 * Detected Mask) / (Sum of Object Mask + 1e-16)
Recall75	Sum of (iou75 * Detected Mask) / (Sum of Object Mask + 1e-16)

[a] IoU = Intersection over Union

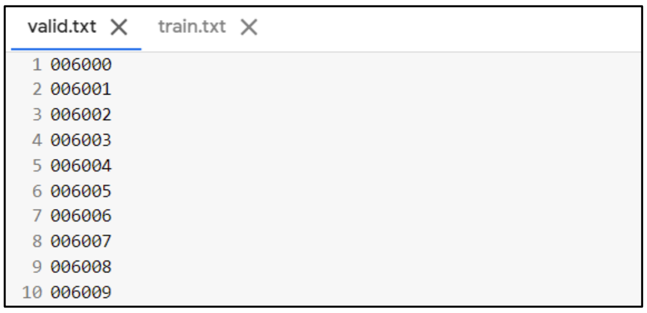
For this study, data is pre-processed and trained under one command. The command for model training is as Figure 3.



```
%cd /content/ObjectDetectionLidarYOLO
!python train.py --epochs 25
```

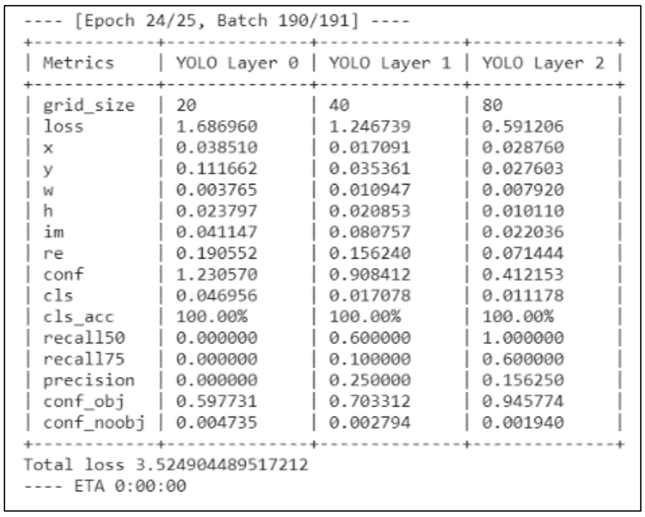
Fig. 3. Training Command

Epoch count for training is set with "--epochs [number of epochs]", while to set the amount of training and testing data, the ImageSets file named train.txt and valid.txt, as shown in Figure 4, is altered manually.



```
006000
006001
006002
006003
006004
006005
006006
006007
006008
006009
```

Fig. 4. ImageSets File



```
---- [Epoch 24/25, Batch 190/191] ----
+-------------+--------------+--------------+--------------+
| Metrics     | YOLO Layer 0 | YOLO Layer 1 | YOLO Layer 2 |
+-------------+--------------+--------------+--------------+
| grid_size   | 20           | 40           | 80           |
| loss        | 1.686960     | 1.246739     | 0.591206     |
| x           | 0.038510     | 0.017091     | 0.028760     |
| y           | 0.111662     | 0.035361     | 0.027603     |
| w           | 0.003765     | 0.010947     | 0.007920     |
| h           | 0.023797     | 0.020853     | 0.010110     |
| im          | 0.041147     | 0.080757     | 0.022036     |
| re          | 0.190552     | 0.156240     | 0.071444     |
| conf        | 1.230570     | 0.908412     | 0.412153     |
| cls         | 0.046956     | 0.017078     | 0.011178     |
| cls_acc     | 100.00%      | 100.00%      | 100.00%      |
| recall50    | 0.000000     | 0.600000     | 1.000000     |
| recall75    | 0.000000     | 0.100000     | 0.600000     |
| precision   | 0.000000     | 0.250000     | 0.156250     |
| conf_obj    | 0.597731     | 0.703312     | 0.945774     |
| conf_noobj  | 0.004735     | 0.002794     | 0.001940     |
+-------------+--------------+--------------+--------------+
Total loss 3.524904489517212
---- ETA 0:00:00
```

The results for the parameter measures in the training phase is taken from the output produced. Figure 5 shows the output for the training phase.

Fig. 5. Training Output

4. Results And Discussion

This section presents the results based on the setup as mentioned in Table 3. Table 5 shows the results of experiment A to find the optimum ratio to split training and testing data for Complex-YOLO. It is seen that a train/test split of 500/500 and 800/200 gave better performance compared to the others. The mAP for 500/500 split is the highest at 56.4%, while the second highest is 800/200 at 51.6%.

Table 5. Experiment (A) Results of Complex-YOLO

Training Data		800	500	700	600
Testing Data		**200**	**500**	**300**	**400**
Accuracy (%)	Layer 1	89	100	100	100
	Layer 2	100	100	100	100
	Layer 3	100	100	100	100
Recall50	Layer 1	0.667	0.600	0.704	0.833
	Layer 2	0.778	0.800	0.889	1.000
	Layer 3	0.889	0.900	0.893	1.000
Recall75	Layer 1	0.111	0.200	0.259	0.250
	Layer 2	0.556	0.300	0.482	0.833
	Layer 3	0.556	0.600	0.607	0.917
Average Precision	Car	0.871	0.864	0.866	0.855
	Pedes -trian	0.441	0.435	0.369	0.315
	Cyclist	0.236	0.393	0.245	0.269
	mAP[a]	0.516	0.564	0.493	0.480

[a.] mAP = Mean Average Precision

Experiment B is done to determine the optimum number of epochs. Due to the limitation in Google Colaboratory, the optimum number is limited to 50 epochs as complications occur if tested with higher epochs for a more extensive set of data for both models. The results, when tested on Complex-YOLO, is shown in Table 6.

Table 6. Experiment (B) Results Of Complex-YOLO

Training Data		800		500	
Testing Data		**200**		**500**	
Epochs		*25*	*50*	*25*	*50*
Accuracy (%)	Layer 1	89	100	100	100
	Layer 2	100	100	100	100
	Layer 3	100	100	100	100
Recall50	Layer 1	0.667	0.625	0.750	0.600
	Layer 2	0.778	1.000	0.750	0.800
	Layer 3	0.889	1.000	0.750	0.900
Recall75	Layer 1	0.111	0.000	0.000	0.200
	Layer 2	0.556	0.625	0.750	0.300
	Layer 3	0.556	0.750	0.500	0.600
Average Precision	Car	0.871	0.788	0.738	0.833
	Pedes -trian	0.441	0.345	0.183	0.254
	Cyclist	0.236	0.171	0.063	0.324
	mAP[a]	0.516	0.435	0.328	0.471

[a.] mAP = Mean Average Precision

It is seen that at 50 epochs for 800/200 splitting, the mAP for the Complex-YOLO model gives a slightly poor score at 43.5% compared to when trained with 25 epochs. However, the class accuracy and recall at 50 epochs is better than the performance at 25 epochs. For the splitting of 500/500, the mAP recorded is 47.1%, and class accuracy is 100% at 50 epochs. At 25 epochs, the mAP is 32.8%, and class accuracy achieved 100%. Thus, it shows that more epochs give better performance for both best split ratios in this study.

Experiment C compares the performance of Complex-YOLO and Tiny-YOLO under the same parameters (epochs was set to 50). Table 7 shows the summary of the comparison between Complex-YOLO and Tiny-YOLO's performance. It is reported that Complex-YOLO has a slightly better performance in terms of precision compared to Tiny-YOLO. However, due to lighter architecture, Tiny-YOLO is seen to withstand more epoch counts in limited time set by Google Colaboratory on a larger amount of data compared to Complex-YOLO.

Table 7. Experiment (C) Results of Complex-YOLO vs Tiny-YOLO

Training Data		800		500	
Testing Data		**200**		**500**	
Model		***Complex-YOLO***	***Tiny-YOLO***	***Complex-YOLO***	***Tiny-YOLO***
Average Precision	Car	0.871	0.817	0.864	0.754
	Pedestr-ian	0.441	0.132	0.435	0.105
	Cyclist	0.236	0.239	0.393	0.181
	mAP[a]	0.516	0.396	0.564	0.347

[a.] mAP = Mean Average Precision

Figure 6 shows the detection results of data trained with Tiny-YOLO and Complex-YOLO, respectively. Objects detected are cars (yellow), pedestrians (red) and cyclists (blue).

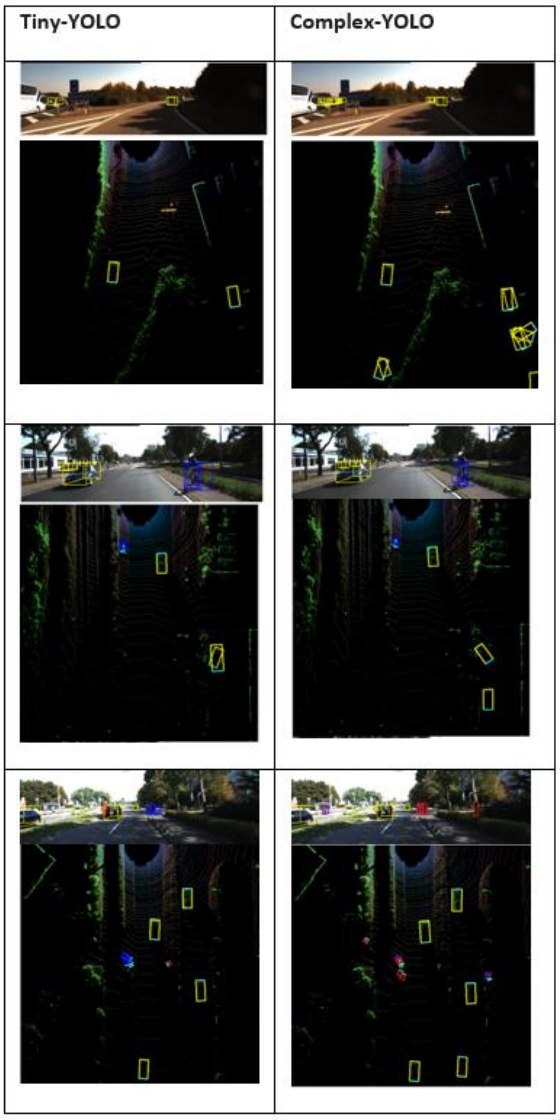


Fig. 6. Detection Results

Based on the results shown in Figure 6, it is observed that the number of bounding boxes by the Tiny-YOLO model appears less than that of Complex-YOLO. This means that Complex-YOLO can detect objects more precise compared to Tiny-YOLO. Both models detected cars precisely, with Complex YOLO being slightly better. This also applies to the detection of cyclists and pedestrians. However, the lack of pedestrians and cyclists in the images contributes to the inadequate results of detection. The capabilities of Tiny-YOLO could be explored more if high-performing hardware was used. Thus, this enables more epochs to be executed.

5. Conclusion

This study has demonstrated the development of YOLO models using LiDAR point cloud data for object detection. Based on the results, the optimum ratio for train/test split is 1:1 (500/500) or 4:1 (800/200), and the optimum number of epochs is limited to 50 due to Google Colaboratory restrictions. The results showed that Complex YOLO has better performance as the mean Average Precision is higher by 0.12 for (800/200) splitting and 0.217 for (500/500) splitting than Tiny-YOLO. Future works that can be done is implementing this study with high-performance hardware. By doing so, more data can be trained with larger epoch counts, hence, the capabilities of Tiny-YOLO can be explored further. Additionally, comparing the results obtained with a 2-stage detector algorithm such as Faster RCNN would further distinguish the contrast between the performances.

Acknowledgment

The authors gratefully acknowledge the use of service and facilities of the Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Shah Alam. This study receives funding from the Ministry of High Education (MOHE), MALAYSIA, under the Fundamental Research Grant Scheme 600-IRMI/FRGS 5/3 (461/2019) of Universiti Teknologi MARA (UiTM).

References

[1] Yahya, M. A., Abdul-Rahman, S., & Mutalib, S. (2020). Object detection for autonomous vehicle with Lidar using deep learning. 2020 IEEE 10th International Conference on System Engineering and Technology, ICSET 2020 - Proceedings, 207–212. https://doi.org/10.1109/ICSET51301.2020.9265358

[2] NTSB. (2018). Preliminary Report HWY18MH010. 4. https://www.ntsb.gov/investigations/AccidentReports/Reports/HWY18MH010-prelim.pdf

[3] Combs, T. S., Sandt, L. S., Clamann, M. P., & McDonald, N. C. (2019). Automated Vehicles and Pedestrian Safety: Exploring the Promise and Limits of Pedestrian Detection. American Journal of Preventive Medicine, 56(1), 1–7. https://doi.org/10.1016/j.amepre.2018.06.024

[4] Roth, M., Jargot, D., & Gavrila, D. M. (2019). Deep End-to-end 3D Person Detection from Camera and Lidar. 2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019, 521–527. https://doi.org/10.1109/ITSC.2019.8917366

[5] Srivastava, S., Divekar, A. V., Anilkumar, C., Naik, I., Kulkarni, V., & Pattabiraman, V. (2021). Comparative analysis of deep learning image detection algorithms. *Journal of Big Data*, *8*(1). https://doi.org/10.1186/s40537-021-00434-w

[6] Mugunthan, N., Sb, B., Harini, C., Naresh, V. H., & V, P. V. (2020). Comparison Review on LiDAR vs Camera in Autonomous Vehicle. IRJET, 07(08), 4242–4246.

[7] Review, A. (2019). Lidar vs. Cameras for Self Driving Cars – What's Best? 1–7. https://www.autopilotreview.com/lidar-vs-cameras-self-driving-cars/

[8] Royo, S., & Ballesta-Garcia, M. (2019). An overview of lidar imaging systems for autonomous vehicles. Applied Sciences (Switzerland), 9(19). https://doi.org/10.3390/app9194093

[9] Borcs, A., Nagy, B., & Benedek, C. (2017). Instant Object Detection in Lidar Point Clouds. IEEE Geoscience and Remote Sensing Letters, 14(7), 992–996. https://doi.org/10.1109/lgrs.2017.2674799

[10] Bagloee, S. A., Tavana, M., Asadi, M., & Oliver, T. (2016). Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. Journal of Modern Transportation, 24(4), 284–303. https://doi.org/10.1007/s40534-016-0117-3

[11] Engels, G., Aranjuelo, N., Arganda-Carreras, I., Nieto, M., & Otaegui, O. (2020). 3D object detection from LiDAR data using distance dependent feature extraction. VEHITS 2020 - Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport Systems, March, 289–300. https://doi.org/10.5220/0009330402890300

[12] Bandyopadhyay, H. (n.d.). YOLO: Real-Time Object Detection Explained. Retrieved August 9, 2021, from https://www.v7labs.com/blog/yolo-object-detection#two-stage-detectors

[13] Simon, M., Milz, S., Amende, K., & Gross, H.-M. (2018). Complex-YOLO: Real-time 3D Object Detection on Point Clouds. 1–14. http://arxiv.org/abs/1803.06199

[14] Choi, J., Chun, D., Kim, H., & Lee, H. J. (2019). Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving. Proceedings of the IEEE International Conference on Computer Vision, 2019-October, 502–511. https://doi.org/10.1109/ICCV.2019.00059

[15] Yin, X., Sasaki, Y., Wang, W., & Shimizu, K. (2020). 3D Object Detection Method Based on YOLO and K-Means for Image and Point Clouds. ArXiv. http://arxiv.org/abs/2005.02132

[16] Adarsh, Pranav & Rathi, Pratibha & Kumar, Manoj. (2020). YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. 687-694. 10.1109/ICACCS48705.2020.9074315.

[17] Benjdira, B., Khursheed, T., Koubaa, A., Ammar, A., & Ouni, K. (2018). Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3. arXiv. https://arxiv.org/abs/1812.10968

[18] Yi, Z., Yongliang, S., & Jun, Z. (2019). An improved tiny-yolov3 pedestrian detection algorithm. Optik, 183, 17–23. https://doi.org/10.1016/j.ijleo.2019.02.038

[19] Gong, X., Ma, L., & Ouyang, H. (2020). An improved method of Tiny YOLOV3. IOP Conference Series: Earth and Environmental Science, 440(5), 052025. https://doi.org/10.1088/1755-1315/440/5/052025

[20] Fang, W., Wang, L., Ren, P., 2020. Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments. IEEE Access 8, 1935–1944.. doi:10.1109/access.2019.2961959

[21] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research. The International Journal of Robotics Research, October, 1–6.

[22] Deepak Ghimire. (2019, December 11). Complex-YOLOv3: PyTorch implementation of Complex-YOLO paper with YoloV3. https://github.com/ghimiredhikura/Complex-YOLOv3