Deep Learning Approaches for Object Detection in Autonomous Driving: Smart Cities Perspective

Othman O. Khalifa[1], Hariz Naufal Mohd Daud[1], Elmustafa Sayed Ali[2,3(✉)], and Mamoon M. Saeed[4]

[1] Department of Electrical and Computer Engineering, Kulliyyah of Engineering, International Islamic University Malaysia, Kuala Lumpur, Malaysia
[2] Department of Electronics Engineering, Sudan University of Science and Technology (SUST), Khartoum, Sudan
elmustafasayed@gmail.com
[3] Department of Electrical and Electronics Engineering, Red Sea University (RSU), Port Sudan, Sudan
[4] Department of Communications and Electronics Engineering, Faculty of Engineering, University of Modern Sciences (UMS), Sana'a, Yemen

Abstract. Object detection has been a key feature of autonomous driving. Autonomous driving is believed to be the solution to the hike in accidents. To develop an object detection model for an autonomous vehicle in smart cities, a few methods were identified by research and studies. Deep learning algorithm that uses artificial neural networks to replace brain functions can perform sophisticated computations on large amounts of data. From the various methods and algorithms available, the performance of each model will vary for each study. This study aims to investigate and identify the best algorithm for detecting objects in smart cities based on deep learning. The chosen algorithm, You Only Look Once (YOLOv5) is then used to build an object detection model with a driving dataset in a framework. The performance of the model trained will then be evaluated and the results will be analyzed. One of the performance evaluation metrics included in this study is the Mean Average Precision (mAP) which will be compared to a few other object detection models.

Keywords: Object detection · autonomous driving · smart cities · deep learning · YOLOv5

1 Introduction

Accidents are highly likely to occur every day in our daily lives, generally more than 80% due to human error according to the Malaysian Institute of Road Safety Research (MIROS) (The Sun Daily, 18, February 2015). As autonomous vehicles are a solution, a connection with smart cities with more sensors and instruments results in smarter

A. Sere et al. (Eds.): AFRICOMM 2023, LNICST 587, pp. 68–80, 2025.
https://doi.org/10.1007/978-3-031-81570-6_5

and more intelligent technology [1]. In a smart city, two-way communication between autonomous vehicles and the data from the cities could vastly improve the current mobility and even the vehicles themselves. However, the challenges of connecting the technologies to replace a function of the human brain with the ability to recognize an object and respond to it is something that needs to be considered [2]. The requirements of autonomous driving to be reliable and accurate in detection and recognition lead towards the deep learning application. Deep learning which consists of an artificial neural network is used to replace the ability of humans to process data, initiate decision-making, create patterns and act as much as a human. The development of deep learning is currently reaching a level where they can learn unsupervised from data that is unlabeled or unstructured [3].

2 Methodology

In this research, the latest object detection algorithm of You Only Look Once (YOLO) was trained to detect vehicles. As a computer vision practitioner, the application of the algorithm, data acquisition and data annotation were the focus.

2.1 Objects Design

There were 12 objects focused on this study. As the model was developed for an autonomous vehicle in smart cities, the objects were related to the surroundings and environment of the vehicle. The objects were bikers, cars, pedestrians, stop signs, traffic lights, green traffic lights, green-left traffic lights, red traffic lights, red-left traffic lights, yellow traffic lights, yellow-left traffic lights, and trucks.

2.2 Proposed Algorithm

The usage of convolutional neural networks was chosen because of their simplicity. A range of pre-trained models that can be modified for a variety of tasks are available. They're also readily available, low-cost to compute, and have acceptable performance characteristics [4]. Object recognition systems from the YOLO family have been demonstrated to outperform other target identification algorithms in vehicle detection tests. YOLOv5 has been shown to improve the processing time of deeper networks significantly. When the project expands to larger datasets and real-time detection, this attribute will become more important [5]. YOLOv5 is written in PyTorch in the initial release instead of PJ Reddie's Darknet. With a better-established ecosystem of PyTorch, support for the implementation is simpler while deployment is easier. Even though Darknet is a flexible research framework, due to a smaller community of users and not being for the production environment, it will create some challenges and less-production ready. Besides that, YOLOv5 is used in this research for its fast and accuracy which is a vital factor for detecting objects for autonomous vehicles instantly and precisely.

For the YOLOv5 model, as it is a single-stage object detector, it has three important parts; refer to Fig. 1 like any other single-stage object detector as below:

1. Model Backbone

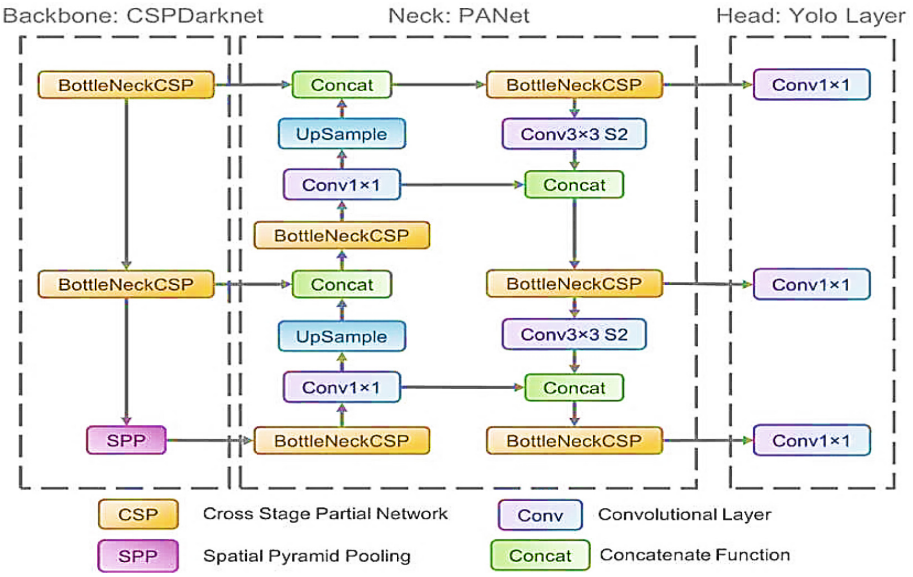

Fig. 1. The network architecture of Yolov5. It consists of three parts: (1) Backbone: CSP Darknet, (2) Neck: PANet, and (3) Head: Yolo Layer.

2. Model Neck
3. Model Head

Model Backbone is mainly used to extract important features from the given input image. In YOLOv5 the CSP—Cross Stage Partial Networks are used as a backbone to extract rich informative features from an input image. CSPNet has shown significant improvement in processing time with deeper networks.

Model Neck is mainly used to generate feature pyramids. Feature pyramids help models to generalise well on object scaling [6]. It helps to identify the same object with different sizes and scales. Feature pyramids are very useful, and help models to perform well on unseen data. Other models use different types of feature pyramid techniques like FPN, BiFPN, PANet, etc. In YOLOv5, the PANet is used as a neck to get feature pyramids. The model Head is mainly used to perform the final detection part. It applied anchor boxes on features and generated final output vectors with class probabilities, objectness scores, and bounding boxes. In YOLOv5 model head is the same as the previous YOLOv3 and YOLOv4 versions [7].

2.3 System Dataset

The Udacity Self Driving Car dataset was used to train YOLOv5, which is a large open-source dataset for object detection, segmentation, and labelling. It is readily available and becomes a choice to save some precious time rather than collecting data and annotating manually [8]. There are nearly 15,000 tagged photos in this collection, with 11 different classes, including cars and trucks. All images are in a resolution of 1920x1200. As a

result, YOLOv5 can be used to detect objects as is, or as a starting point for an adjusted model to detect features such as the front and back of the objects.

2.4 Adopted Framework and Tools

Google Colab or Collaboratory, a popular product of Google research provides access for anyone to write and execute arbitrary Python code just by using the browser. It is a purposely built environment and well suited for machine or deep learning, data analysis as well and education. This free-to-use environment is a Jupyter Notebook environment that requires no setup and runs entirely in the cloud. Artificial intelligence and deep learning are a long-life study [9]. The researchers develop and train their models using the Python language. Therefore, all codes were done in Google Colab and were written using Python.

PyTorch is a framework that has two data primitives that enable work with both pre-loaded datasets and your data. The samples and their labels are stored in Dataset, and Data Loader wraps an expected around the Dataset to make it easy to access the samples. To load the Udacity dataset into the environment, PyTorch was used.

Lastly, the visualization tool to provide the measurements and visualizations needed for the deep learning workflow used was Tensor Board. The tool allows the tracking of performance metrics of the model such as accuracy and loss, visualizing the model graph as well as projecting embedding's to a lower-dimensional space [10].

2.5 Training of the Proposed Model

Before the training of the model started, the YOLOv5 repository by Ultralytics Company was cloned into the environment by a few lines of code. Once the cloning process was done, the dependencies such as torch and IPython were installed. Next, the dataset was downloaded and loaded in PyTorch format from the Roboflow website including the YAML file defining the test and train data location. The YAML file then writes the parameters like the number of classes, anchors and each layer [11]. When everything was successfully prepared, the training was initiated at an initial of 50 epochs and gradually increased up to 200 epochs. The training took some time to complete as it depends on the resources of the Graphic Processing Unit (GPU) and Random-Access Memory (RAM) from Google Colab.

2.6 Evaluating the Proposed Model

After a few hours, the training stopped at 200 epochs and Tensor-Board was launched to visualize the performance metrics of the model. A few graphs were shown indicating the Mean Average Precision (mAP), Precision and Recall values as well as losses in box, classification and object. Besides that, the ground truth of the data also was checked to identify potential flaws in the data [12]. From the trained weights, a few video inputs were run as the inference or test for the model. The results will be discussed further in the next section.

3 Results and Discussion

The object detection model was initially trained for 20 epochs and gradually increased over 200 epochs for a better result. It took approximately 3 h and 17 min, efficiently making use of a Python 3 Google Compute Engine Backend (GPU) computing power provided by the Google Colab. During each epoch, the confidence metric improved over time, making detection closer to perfect [13, 14]. Once the training starts, the training images, labels and augmentation effects can be visualized for the ground truth to be observed. Below is the figure; refer to Fig. 2 showing the ground truth training data for the model.

The next figure; refers to Fig. 3 showing the augmented training example from our dataset. The augmentation was done during the pre-processing of the dataset which has the objective of preventing the YOLOv5 network from learning irrelevant patterns from our dataset and essentially boosting the overall performance. It will add slightly modified copies of already existing data to increase the amount of data then lead to a better prediction accuracy [15].

Fig. 2. Ground truth training data

Through this visualization, it is a vital process to both identify potential flaws in the ground truth labels, as well as to look for insights that will guide the model development. Thus, visualizing the training data is the foundation of every successful deep-learning project [16]. Once the training ended with the visualization and validation of ground truth training data, the performance of the model was evaluated by running some code to launch the Tensorboard. Training losses and performance metrics were saved to Tensorboard and a log file when we trained the model. The results file was plotted as a PNG file after training was completed.

The first metric was the Mean Average Precision (mAP) graph which is a popular metric in measuring the accuracy of a model. It does compute the average precision value for recall value over 0 to 1. Precision measures how accurate are the predictions while Recall measures how well the detector finds all the positives. From the figure below;

Fig. 3. Ground truth augmented training data

refer to Fig. 4, the mAP for intersection over Union (IoU) threshold of 0.5 obtained was initially at 25.08% for 20 epochs training. For 200 epochs, the mAP valued at 60.36% which indicates that the accuracy improves a lot for this model.

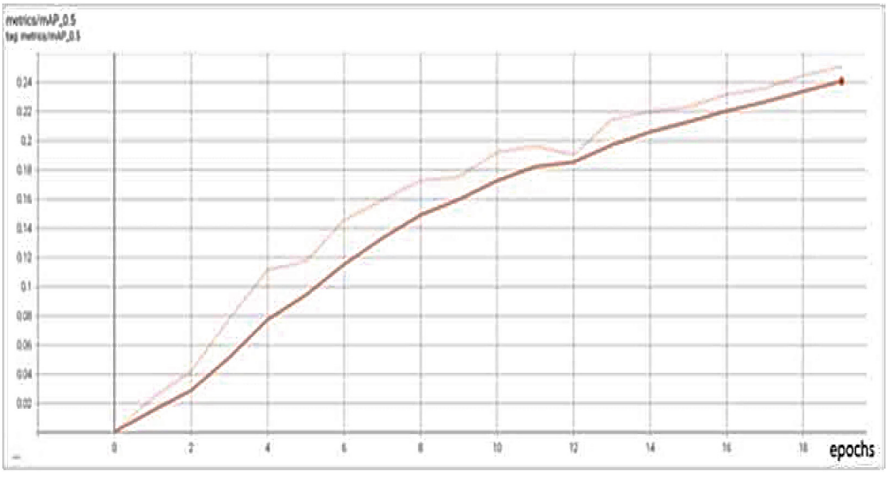

Fig. 4. mAP for IoU of 0.5 for 20 epochs

While for the mAP for the IoU threshold of 0.5 to 0.95; refer to Fig. 5 and Fig. 6, it gave a value of 11.36% for 20 epochs and 30.64% for 200 epochs training. From both mAP graphs, it can be observed that the 200 epochs training showed a smoother and linearly interpolated graph which indicates a better performance of the model. From the precision graphs below; refer to Fig. 7 and Fig. 8, it can be observed that 200 epochs

training was more precise in every epoch. However, the graph was a little bit overfitted as the dataset is quite small compared to the training epochs. To overcome this, more data will be needed and prepared in the custom driving dataset in Kuala Lumpur.

Meanwhile, the Recall curves; refer to Fig. 9 and Fig. 10 also seem a little bit overfitted for 200 epochs but overall, the recall curve of 200 epochs was better than the 20 epochs curve.

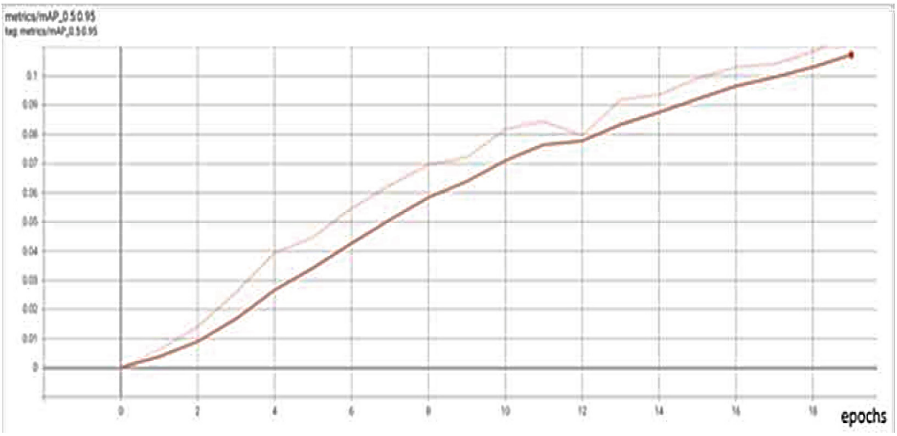

Fig. 5. mAP for IoU of 0.5 to 0.95 for 20 epochs

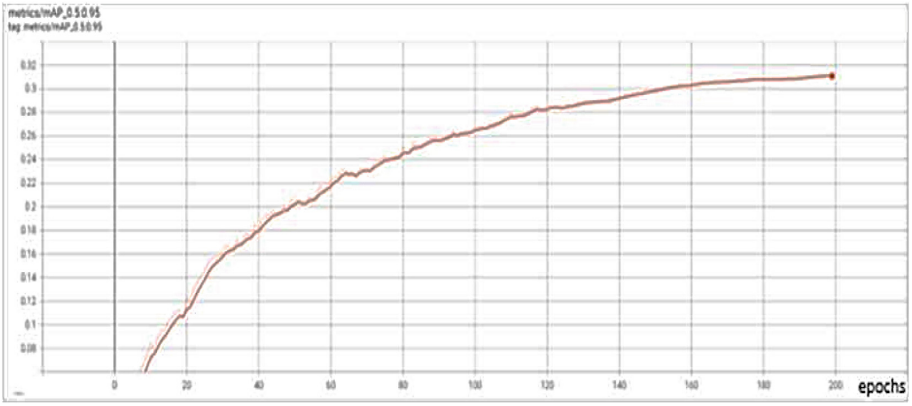

Fig. 6. mAP for IoU of 0.5 to 0.95 for 200 epochs

Comparing both figures, the bounding box loss, classification loss as well and object loss improve a lot as the loss value decreases a lot; refer to Fig. 11 and Fig. 12. The box loss indicates how well the algorithm can detect the centre of the object and how well the predicted bounding box encompasses it. Objectness is a probability measure for the presence of an object in a proposed region of interest. If the objectivity is high, the image window is likely to have an object in it. The algorithm's classification loss indicates how

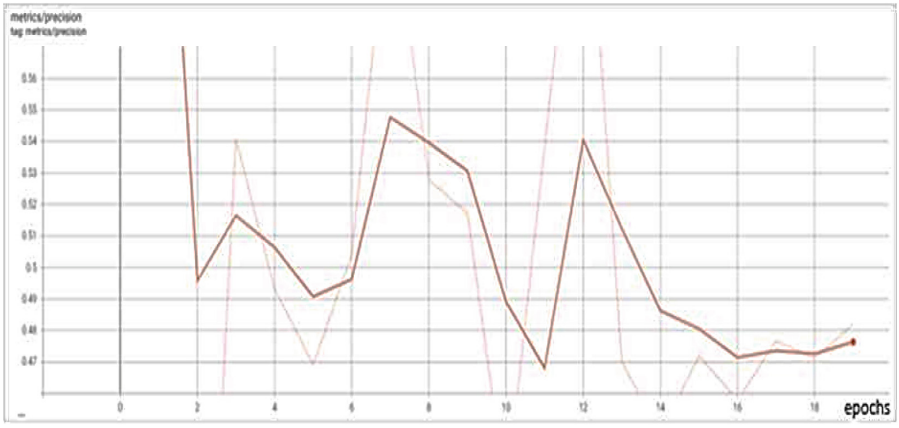

Fig. 7. Precision values for 20 epochs

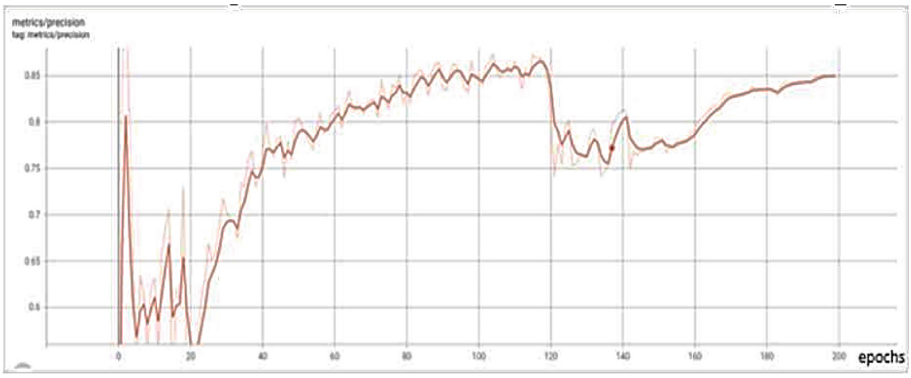

Fig. 8. Precision values for 200 epochs

effectively it can predict the proper class of a given object (Kasper-ulcers et al., 2021). Precision, recall, and mean average precision all improved rapidly before plateauing after roughly 150 epochs. The validation data's box, objectness, and classification losses likewise showed a significant drop until around 150 epochs.

The evaluation on YOLO using real-time data sets is summarized through an example in Fig. 12 and Fig. 13. Here in the first figure; refer to Fig. 13, the biker was successfully detected but it also detects it as a car. As for the cars, some of the cars were undetected by the model. However, in the second trained model; refer to Fig. 14, all the cars and the bikers were successfully correctly detected and true positive. This shows that the accuracy improves and it benefits the purpose of this project to detect objects for autonomous driving.

Based on the evaluation of the models trained, the model with better mAP, Precision and Recall value as well as lower losses of box, classification and object should be used

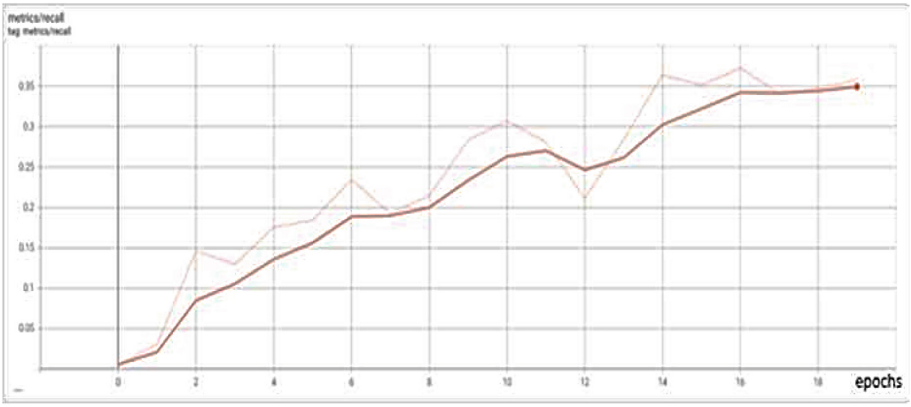

Fig. 9. Recall values for 20 epochs

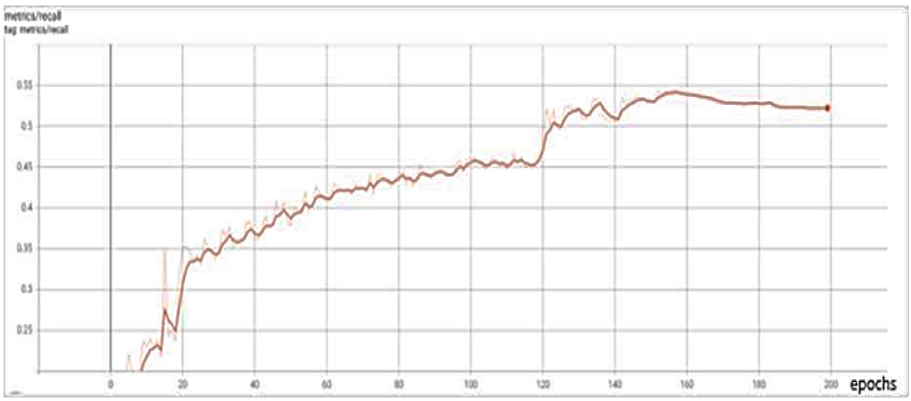

Fig. 10. Recall values for 200 epochs

for object detection model for autonomous driving [17]. With the trained and evaluated model, the object detection model was then compared to some existing solutions (Faramarzi, 2020) [18]. The table below shows the comparison with a model trained for autonomous driving using YOLO. From Table 1 above, it can be concluded that the YOLOv5 network in this study performs well and obtained a better Mean Average Precision compared to the other models trained. Even though the result obtained was improved, room for improvement is still available to increase the accuracy of the model. Besides that, the FPS for the YOLOv5 object detection model was higher than the others. From this comparison, YOLOv5 is a model that is suited for an autonomous vehicle.

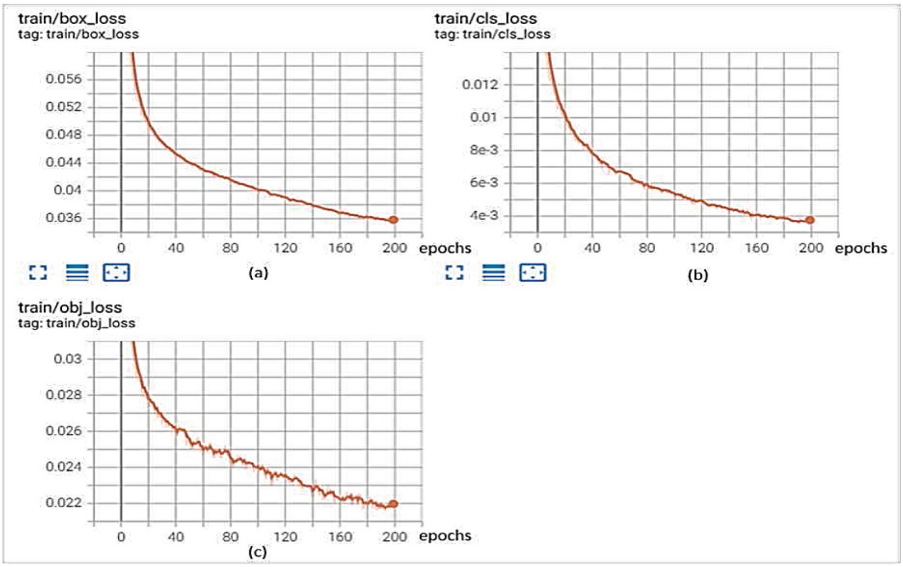

Fig. 11. Losses metrics for 20 epochs

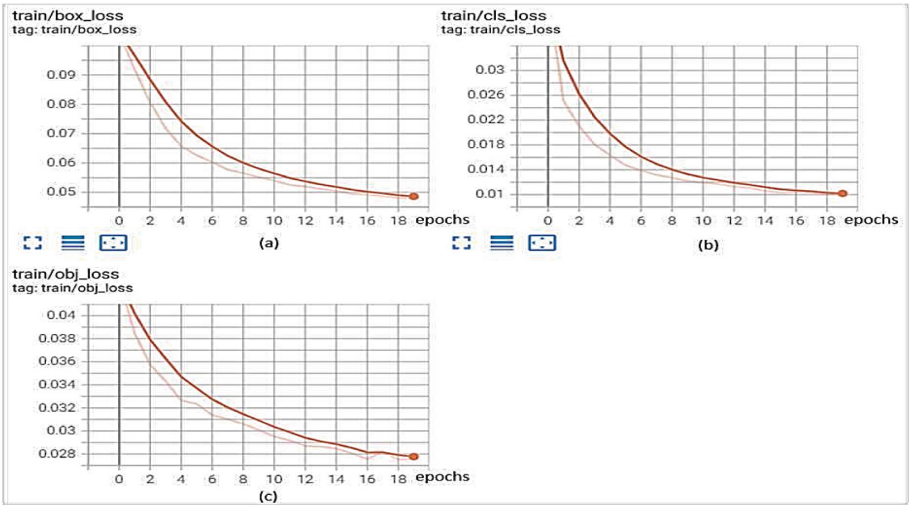

Fig. 12. Losses metrics for 200 epochs

Fig. 13. Inference on 20 epoch's model

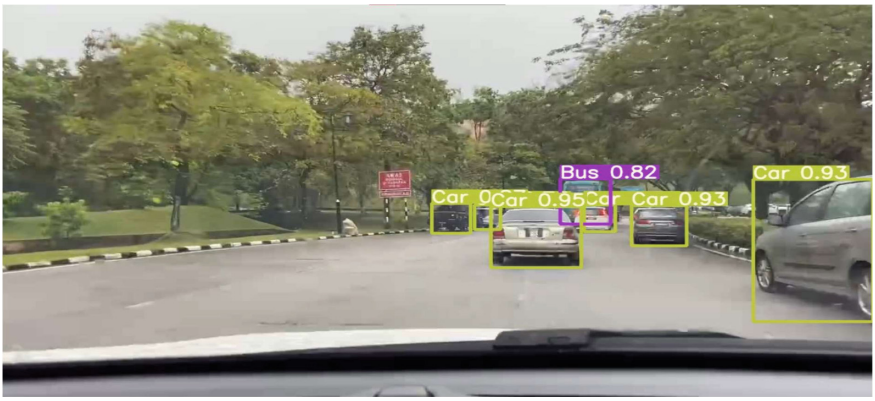

Fig. 14. Inference on 200 epoch's model

Table 1. Comparison of YOLO object detection model

Model	Dataset	mAP	FPS
YOLOv2- 608x608	COCO	48.1	40
YOLOv3–608	COCO	57.9	20
YOLOv3	Udacity	28.19	32
YOLOv4	Udacity	41.46	45

4 Conclusion

YOLO is announced as one of the most promising detection systems using neural networks. This study investigates the best algorithm and the development of the object detection model for an autonomous vehicle in smart cities. By performing a comprehensive analysis of the YOLOv5 network over the Udacity dataset, it was discovered that YOLO can achieve 61% precision with 50% recall at 58 frames per second. The results

are encouraging and suggest that YOLO is an excellent model for detecting objects required for autonomous driving systems. Further improvements may be made from time to time following the growth of the technology in this decade and improve the speed and accuracy.

References

1. Alatabani, L.E., Ali, E.S., Saeed, R.A.: Deep learning approaches for IoV applications and services. In: Magaia N., Mastorakis G., Mavromoustakis C., Pallis E., Markakis E.K. (eds.) Intelligent Technologies for Internet of Vehicles. Internet of Things (Technology, Communications, and Computing). Springer, Cham (2021). https://doi.org/10.1007/978-3-030-76493-7
2. Ali, E.S., Hassan, M.B., Saeed, R.A.: Machine learning technologies on internet of vehicles. In: Magaia N., Mastorakis G., Mavromoustakis C., Pallis E., Markakis E.K. (eds.) Intelligent Technologies for Internet of Vehicles. Internet of Things (Technology, Communications, and Computing). Springer, Cham (2021). https://doi.org/10.1007/978-3-030-76493-7
3. Alqurashi, F.A., Alsolami, F., Abdel-Khalek, S., Sayed Ali, E., Saeed, R.A.: 'Machine learning techniques in internet of UAVs for smart cities applications'. J. Intell. Fuzzy Syst. **42**(4), 1–24 (2021). https://doi.org/10.3233/JIFS-211009
4. Arie, L.G., PhD.: The practical guide for Object Detection with YOLOv5 algorithm. Medium (2022, April 1). https://towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843
5. Khan, A., et al.: PackerRobo: model-based robot vision self-supervised learning in CART. Alexandria Eng. J. **61**(12), 12549−12566 (2022).https://doi.org/10.1016/j.aej.2022.05.043
6. Hassan, M. B., Ahmed, E S., Saeed, R.A.: Machine learning for industrial IoT systems. In: Zhao, J., Vinoth Kumar, V. (eds.) Handbook of Research on Innovations and Applications of AI, IoT, and Cognitive Technologies, pp. 336–358. Hershey, PA: IGI Global, 2021. https://doi.org/10.4018/978-1-7998-6870-5.ch023
7. Xu, Q., Zhu, Z., Ge, H., Zhang, Z., Zang, X.: Effective face detector based on YOLOv5 and superresolution reconstruction. Comput. Math. Methods Med. **16**(2021), 7748350 (2021). https://doi.org/10.1155/2021/7748350.PMID:34824599;PMCID:PMC8610656
8. Taye, M.M.: Understanding of machine learning with deep learning: architectures, workflow, applications and future directions. Computers **12**, 91 (2023). MDPI
9. Elmustafa, S.A., et al.: machine learning technologies for secure vehicular communication in internet of vehicles: recent advances and applications. Wiley-Hindawi, J. Secur. Commun. Netw. (SCN) **2021**, 8868355 (2021), https://doi.org/10.1155/2021/8868355
10. Faramarzi, M.: Road Damage Detection and Classification Using Deep Neural Networks (YOLOv4) with Smartphone Images (2020). https://doi.org/10.13140/RG.2.2.21734.65602
11. Gadal, S., Mokhtar, R., Abdelhaq, M., Alsaqour, R., Ali, E.S., Saeed, R.: Machine learning-based anomaly detection using K-mean array and sequential minimal optimization. Electronics **11**, 2158 (2022). https://doi.org/10.3390/electronics11142158
12. Hameed, S.A., et al.: Framework for enhancement of image guided surgery: Finding area of tumor volume. Aust. J. Basic Appl. Sci. **6**(1), 9–16 (2012)
13. Kasper-Eulaers, M., Hahn, N., Berger, S., Sebulonsen, T., Myrland, Ø., Kummervold, P.: Short communication: detecting heavy goods vehicles in rest areas in winter conditions using YOLOv5. Algorithms. **14**, 114 (2021). https://doi.org/10.3390/a14040114
14. Khalifa, O.O., et al.: An IoT-platform-based deep learning system for human behavior recognition in smart city monitoring using the Berkeley MHAD datasets. Systems **10**, 177 (2022). https://doi.org/10.3390/systems10050177

15. Ahmed, K.E.B., Mokhtar, R.A., Saeed, R.A.: A new method for fast image histogram calculation. In: International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), pp. 187−192, Khartoum, Sudan (2015)
16. Anatabine, L.E., Elmustafa, S.A., Mokhtar, R.A., Saeed, R.A., Alhumyani, H., Hasan, M.K.: Deep and reinforcement learning technologies on internet of vehicle (IoV) applications: current issues and future trends. J. Adv. Trans. **2022**, Article ID 1947886, 16 (2022). https://doi.org/10.1155/2022/1947886
17. Mansour, R.F., Alfar, N.M., Abdel-Khalek, S., Abdelhaq, M., Saeed, R.A., Alsaqour, R.: Optimal deep learning-based fusion model for biomedical image classification. Expert Syst. **39**(1), 34–54 (2021). https://doi.org/10.1111/exsy.12764
18. Faramarzi, M.: Road damage detection and classification using deep neural networks (YOLOv4) with smartphone images (June 15, 2020). Available at SSRN: https://ssrn.com/abstract=3627382. https://doi.org/10.2139/ssrn.3627382