Deep Learning-Based Vehicle Type Detection and Classification.

NADIN PETHIYAGODA, MWP MADURANGA, DMR KULASEKARA, TL WEERAWARDANE
Department of Computer Engineering
General Sir John Kotelawala Defence Univerity
10390, Ratmalana,
SRI LANKA

Abstract: - Modern intelligent transportation systems heavily rely on vehicle-type classification technology. Deep learning-based vehicle-type categorization technology has sparked growing concern as image processing, pattern recognition, and deep learning have all advanced. Over the past few years, convolutional neural work, in particular You Only Look Once (YOLO), has shown to have considerable advantages in object detection and image classification. This method speeds up detection because it can predict objects in real time. High accuracy: The YOLO prediction method produces accurate results with low background errors. YOLO also understands generalized object representation. This approach, which is among the best at object detection, outperforms R-CNN approaches by a wide margin. In this paper, YOLOv5 is used to demonstrate vehicle type detection; the YOLOv5m model was chosen since it suits mobile deployments, the model was trained with a dataset of 3000 images, where 500 images were allocated for each class with a variety of vehicles. Hyperparameter tuning was applied to optimize the model for better prediction and accuracy. Experimental results for a batch size of 32 trained for 300 epochs show a precision of 98.2%, recall of 94.9%, mAP@.5 of 97.9%, mAP@.5:.95 of 92.8%, and overall accuracy of 95.3% trained and tested on four vehicle classes.

Key-Words: - You Only Look Once (YOLO), Deep Learning, Convolutional Neural Networks (CNN), Single Shot Detector (SSD), Vehicle Detection, Vehicle Recognition

Received: June 7, 2022. Revised: March 16, 2023. Accepted: May 12, 2023. Published: June 2, 2023.

1 Introduction

Various advancements in the field of machine vision have fundamentally transformed the world. Technology has had an impact on various industries, including transportation. Because of population increase and human requirements, the use of vehicles has risen dramatically. As a result of the increased difficulties in controlling these vehicles, Intelligent Traffic Systems were developed, Vehicle Type Detection systems are critical components of intelligent traffic systems, and they have a wide range of applications [17], including highway toll collection, traffic flow statistics, and urban traffic monitoring. The development of autonomous driving technology has given people a new knowledge of high-level computer vision, and intelligent transportation and driverless driving technologies have drawn an increasing amount of interest. Vehicle Type Detection is a relatively significant technology in intelligent transportation and autonomous driving. Thanks to the rapid development of large-scale data, computer hardware, and deep learning technologies, there are now several techniques for classifying various types of vehicles. Most methodologies that can be implemented have been employed by CNN, Faster RCNN, YOLO, and SSD [2,18]. The shortcomings of current object detection systems are addressed in this work using the most recent real-time object detection method, namely YOLO.

A. YOLO

You only look once (YOLO) is a state-of-the-art, real-time object detection system introduced in 2015 by [13]. YOLO proposes using an end-to-end neural network that provides predictions of bounding boxes and class probabilities all at once as opposed to the strategy used by object detection algorithms before it, which repurpose classifiers to do detection. R-CNNs are a type of two-stage detector and one of the early deep learning-based object detectors. The major issue with the R-CNN family of networks is their speed. However, though they frequently produce very accurate results, they were incredibly slow, averaging barely 5 FPS on a GPU. YOLO employs a one-stage detector technique to aid in accelerating deep learning-based object detectors. YOLO has the natural advantage of speed, better Intersection over Union in bounding boxes, and improved prediction accuracy compared to real-time object detectors. YOLO runs at up to 45 FPS, making it a far faster algorithm than its competitors. The GoogleNet architecture inspired YOLO's architecture, YOLO's architecture has a total of 24 convolutional layers with 2 fully connected layers at the end. The main

problems with YOLO, the identification of small objects in groups and the localization accuracy—were supposed to be addressed by YOLOv2 [14]. By implementing batch normalization, YOLOv2 raises the network's mean Average Precision. The addition of anchor boxes, as suggested by YOLOv2, was a considerably more significant improvement to the YOLO algorithm. As is well known, YOLO predicts one object for every grid cell. Although this simplifies the constructed model, it causes problems when a single cell contains several objects because YOLO can only assign one class to the cell.

YOLOv2 removes this restriction by allowing the prediction of several bounding boxes from a single cell. The network is instructed to anticipate five bounding boxes for each cell to do this. YOLO9000 [14] was presented as a technique to discover more classes than COCO as an object detection dataset could have made possible, using a similar network design to YOLOv2. Although YOLO9000 has a lower mean Average Precision than YOLOv2, it is still a powerful algorithm because it can identify over 9000 classes. YOLOv3 [15] was proposed to enhance YOLO with modern CNNs that utilize residual networks and skip connections. YOLOv2 employs DarkNet-19 as the model architecture, but YOLOv3 uses the significantly more intricate DarkNet-53, a 106-layer neural network with residual blocks and up-sampling networks, as the model backbone. With the feature maps being extracted at layers 82, 94, and 106 for these predictions, YOLOv3's architectural innovation allows it to forecast at three different sizes.

YOLOv4 [16] is built using CSPDarknet53 as the backbone, SPP (Spatial pyramid pooling), and PAN (Path Aggregation Network) for what is known as "the Neck," and YOLOv3 for "the Head" following recent research findings. This system uses the latest algorithm, YOLOv5, which uses the PyTorch framework possessing many advantages such as smaller size, higher performance, and better integration than YOLOv4.

2 Related Works

In the works of [2][6], the authors have presented experimental results that that YOLOv4 had better performance, F1 score, precision, recall, and mAP values compared to other models in [2] and YOLOv3 has demonstrated better results in performance and accuracy than R-CNN and Fast R-CNN [6]. Yanhong Yang [3] uses the SSD algorithm to achieve vehicle classification and positioning, from picture collection, picture calibration, model training, and model detection, several aspects of the detailed introduction of the vehicle classification process. THE PASCAL VOC dataset was used, and the TensorFlow framework and SSD model with the VGG16 model were used for model training. In [2-9] [11][12] Common vehicle categories are bus, car, truck, bus, and motorbikes. In [6-7] limitations were how to effectively detect vehicles in complex environments. Due to the limitations of hardware and time, in-depth research can be conducted in the future on the aspects of improving accuracy, improving detection accuracy, and improving calibration methods. A combination of YOLOv4 and DeepSORT has been used in [7] for vehicle detection and real-time object tracking, respectively.

In [8] proposes a CNN model for vehicle classification with low-resolution images from a frontal perspective. The model was trained as a multinomial logistic regression where the cross-entropy of the ground truth labels and the model's prediction estimate the error. Data augmentation was performed to prevent overfitting. A leaky rectifier activation function (LReLU) instead of (ReLU) was set up for the convolution output. However, [10] proposed a CNN architecture for vehicle type classification. The system requires only one input, a vehicle image. The model consists of two convolution layers, 1st, and 2nd layers. Two pooling layers and four activation functions (ReLU) The 3rd, 4th, and 5th layers are fully connected. In [12] proposed the network developed has a total of 13 layers, 1 convolutional input layer, 11 intermediate layers including a combination of Rectified Linear Unit (ReLU) activation, convolutional, dropout, max-pooling, flatten, and densely-connected layers, and 1 Softmax output layer. In the works [6][11] the gathered datasets from public sources such as COCO, OpenImage, PASCAL VOC, and some works their traffic data collected from camera sources. The dataset split was 80:20 80% for training and 20% for testing [6][9].

In the works [1][12] The test data gave it had produced better accuracies with pictures with high definition while for the pictures with low definition, the recognition accuracy decreases. It is also observed that the probability of identifying small cars as medium-sized vehicles is only 8.69%, and the probability of identifying large cars is lower, 2.14% only in [1] and. Further improvements in prediction accuracy include training on more quality images to allow it to extract more features from the data and further divide it into more classes [12]. In [5][10] the authors wish to aim for better accuracies and stability by searching for suitable hyperparameters. Research gaps in [5-7] show the need to cover more variations of vehicles, Cars Image datasets need more data to classify, train, and real-time data analysis of the

traffic and also more complex environmental conditions such as night-time and heavy rain. In [8-9][11] images that could be produced were replicated using data augmentation to improve precision and [4] stated image processing techniques were used to improve prediction accuracy. In [11] the authors have used Faster R-CNN, for shareable convolutional layers of RPN and detection network, the improved ZF net is applied on the PASCAL VOC2012 as the backbone network.

In [12] the authors have developed a CNN, to detect types of vehicles commonly found on the road for database collection purposes and improve the existing vehicle recognition for advanced applications. [10] To avoid overfitting Dropout method has been used, and the final layer is the predictor. TensorFlow was used to implement the CNN structures. The hyperparameters for the CNN model were also mentioned which can affect the performance of the CNN. The dataset mentioned was obtained from extracted frames of a video source. In works [11] RPN is trained by using Stochastic Gradient Descent (SGD). The method has better detection average precision for cars and trucks, while the average precision of minivans and buses is lower. The result might be caused by a little training set of minivans and buses.

Some knowledge gaps were identified in the literature review conducted; many authors have included foreign datasets, resulting in fewer accuracies when tested over real-time data. Some researchers are trying to improve the performance and recognition accuracies by considering images of different lighting, weather conditions, noise reduction, etc. which had been a challenge. According to the works in [2], it is clear that YOLO had outperformed other models such as Faster-RCNN and SSD.

3 Framework Design and Methodology

This section outlines the methodology employed for collecting and constructing the dataset required for model training and testing. It also delves into the exploration and comprehension of the architecture underlying YOLOv5, while determining the optimal approach for achieving the desired output.

3.1 Dataset

In this experiment, the gathered dataset was from public sources such as Kaggle, Stanford Cars Dataset, vehicle images scraped from stock photo websites, and traffic data collected from a video source with the following characteristics: Video duration: 300 seconds, resolution: 1080x2340 pixels, frame rate: 30 FPS. The dataset was prepared for six major types of vehicles: car, three-wheel, bus, truck, motorbike, and van. These images are of different illumination, angle, and different vehicle models. The total dataset size is 3,000 images of different resolutions, with 500 images allocated for each class. The gathered dataset was annotated in YOLO format using a free open source called Labelling to graphically label the images. For each image file in the same directory, a text file with the same name is created in the YOLO labeling format. Each text file provides the object class, object coordinates, height, and width for the accompanying image file. The dataset was split into train and valid, which are 70% (2,100 images) and 30% (900 images) respectively

Fig. 1. Sample images from the training dataset

B. YOLOv5

The most cutting-edge object detection algorithm currently in use is the YOLOv5 [19], which Ultralytics introduced in June 2020. It is a novel convolutional neural network (CNN) that accurately detects objects in real-time. This method processes the entire image using a single neural network, then divides it into parts and forecasts bounding boxes and probabilities for each component. The predicted probability is used to weigh these bounding boxes. In the sense that it only performs one forward propagation cycle through the neural network, the approach "only looks once" at the image before making predictions. After non-max suppression, it then provides discovered items. YOLOv5 consists of:

- Backbone: New CSP-Darknet53
- Neck: SPPF, New CSP-PAN
- Head: YOLOv3 Head

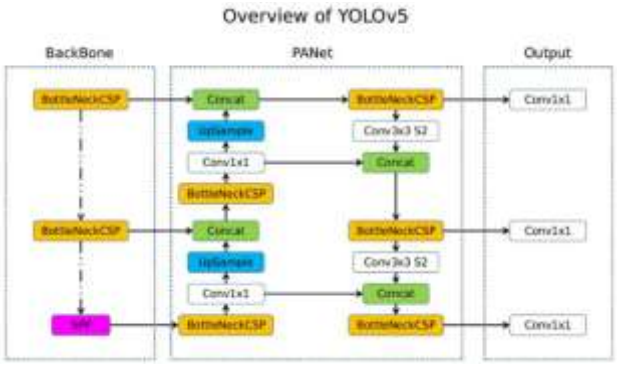

Fig. 2. YOLOv5 architecture [19]

The overview of the YOLOv5 architecture is shown in Fig. 2. To understand the classes of objects in the data, YOLOv5 models need to be trained using labeled data. The custom dataset that was prepared was in the YOLO format with one text file per image. The text file specifications are:

- One row per object
- Each row is in class x_center y_center width height format.
- Box coordinates must be in normalized xywh format (from 0 - 1). If your boxes are in pixels, divide x_center and width by image width, and y_center and height by image height.
- Class numbers are zero-indexed (start from 0).

YOLOv5 provides pre-trained models:

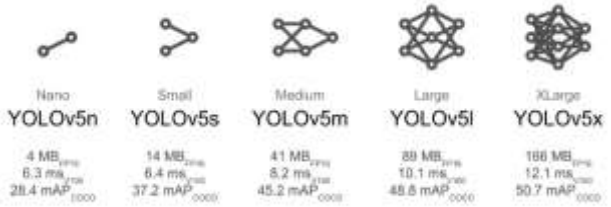

Fig. 3. YOLOv5 models [19]

Larger models, such as YOLOv5x and YOLOv5x6, will almost always yield better results, but they contain more parameters, need more CUDA memory to train, and run more slowly.

The YOLOv5 loss consists of three parts:

- Classes loss (BCE loss)
- objectless loss (BCE loss)
- Location loss (CIoU loss)

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \quad (1)$$

The abjectness losses of the three prediction layers (P3, P4, P5) are weighted differently. The balance weights are [4.0, 1.0, 0.4] respectively.

$$L_{obj} = 4.0 \cdot L_{obj}^{small} + 1.0 \cdot L_{obj}^{medium} + 0.4 \cdot L_{obj}^{large} \quad (2)$$

YOLOv5 uses the following formula to calculate the predicted target information:

$$b_x = (2 \cdot \sigma(t_x) - 0.5) + C_x \quad (3)$$

$$b_y = (2 \cdot \sigma(t_y) - 0.5) + C_y \quad (4)$$

$$b_w = p_w \cdot (2 \cdot \sigma(t_w))^2 \quad (5)$$

$$b_h = p_h \cdot (2 \cdot \sigma(t_h))^2 \quad (6)$$

The build targets to match positive samples: Calculate the aspect ratio of GT and Anchor Templates.

$$r_w = w_{gt}/w_{at} \quad (7)$$

$$r_h = h_{gt}/h_{at} \quad (8)$$

$$r_w^{max} = \max(r_w, \frac{1}{r_w}) \quad (9)$$

$$r_h^{max} = \max(r_h, \frac{1}{r_h}) \quad (10)$$

$$r^{max} = \max(r_w^{max}, r_h^{max}) \quad (11)$$

$$r^{max} = anchor_t \quad (12)$$

The final metric, the mAP across test data, is generated by averaging all mAP values for each class in order to determine the mAP across various Intersection over Union (IoU) thresholds.

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \quad (13)$$

AP_k = the Average Precision of a class k

n = the number of classes

4 Model Training and Evaluation

3.1 Initial Training Results

The model was trained on a system equipped with Ubuntu 20.04.3 LTS, CUDA 11.4, 32 GB RAM, NVIDIA GeForce RTX 3090, Python 3.9.12, PyTorch 1.12.0. The yolov5m model was used for the training and test purpose, and a YAML file was defined to configure the paths for the dataset and the number of classes to train (car, three-wheel, bus, truck, motorbike, and van). The model was trained for 300 epochs with a batch size of 32, and (The weights and Bias) Platform was used to visualize and track data in real time.

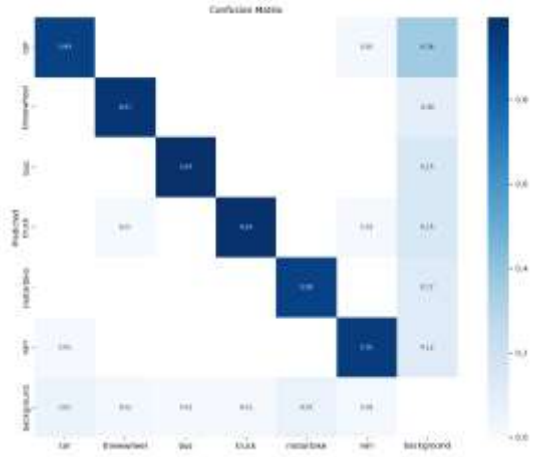

Fig.4(a)

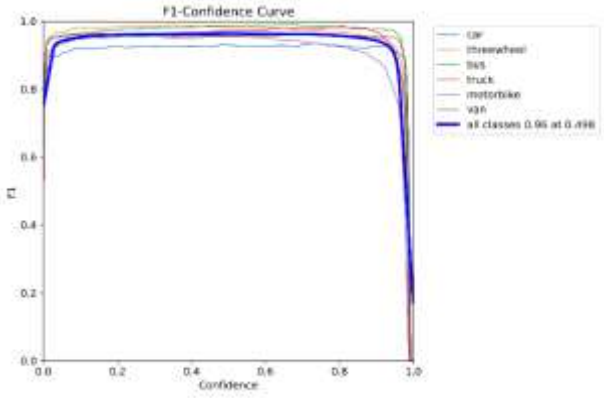

Fig.4(b)

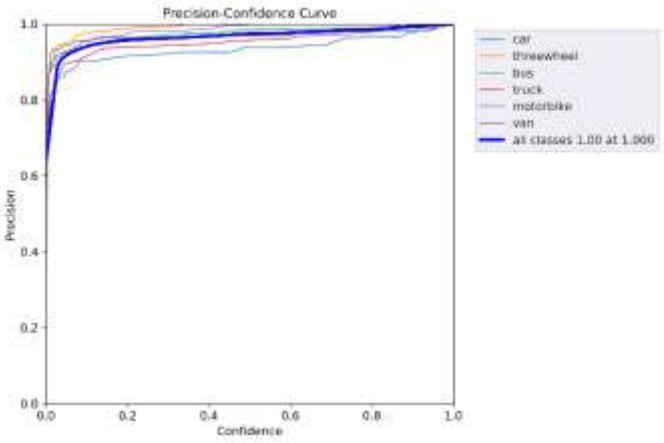

Fig.4(c)

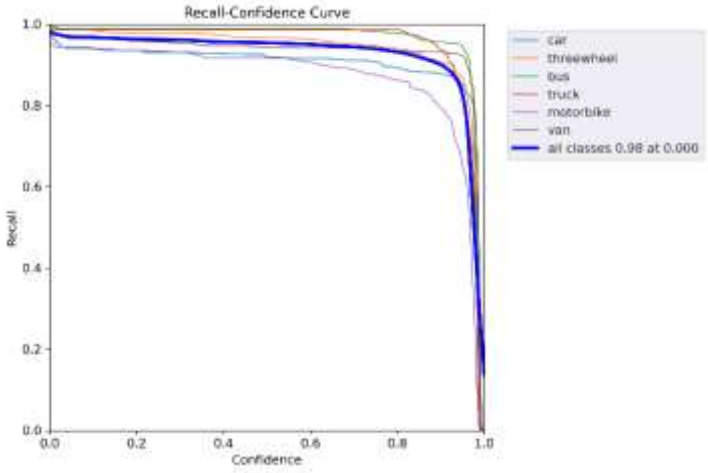

Fig.4(d)

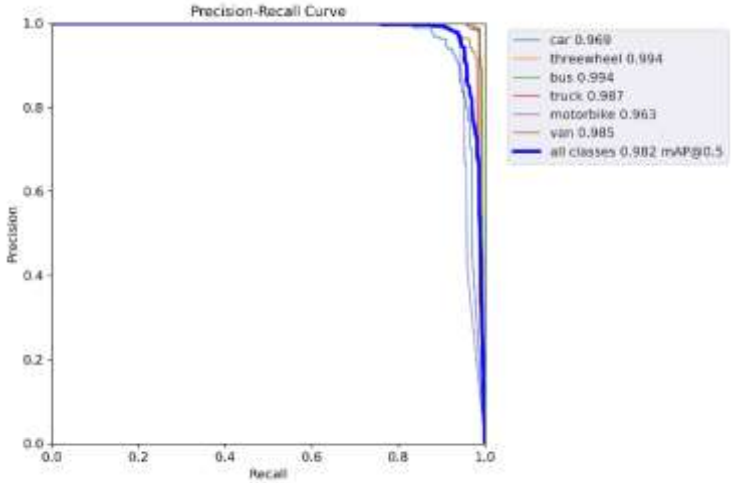

Fig.4 (e)

Nadin Pethiyagoda, Mwp Maduranga,
Dmr Kulasekara, Tl Weerawardane

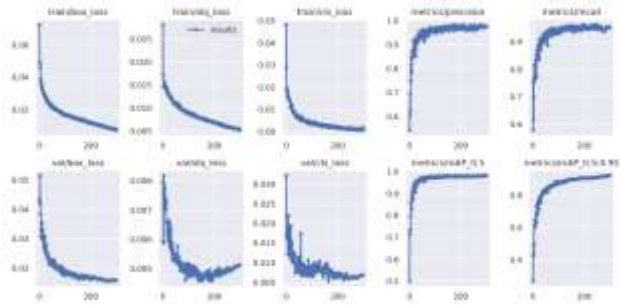

Fig.4(f)
Fig. 4. Training results

Fig. 4 (a) shows the confusion matrix, 93% of samples were predicted as True Positives (TP) for car, 97% for three-wheel, 99% for both bus and truck, 93% for motorbike, and 95% for van whereas 1% of the truck and 2 % of the car were confused. This is due to when viewed from the front, a car and a truck both have common details. Fig.4 (b) shows the weighted harmonic mean of a classifier's precision (P) and recall (R), considering = 1, which is known as the F-measure (F1 score). The F1-Confidence score was obtained as 0.96 at 0.498 confidence. Fig. 4 (c) presents the precision of each class obtained after the completion of training; it is the proportion of correct identifications. The Precision-Confidence score was obtained as 1.00. Fig. 4 (d) presents the recall of each class; it is the proportion of actual positives which are identified correctly. The Recall-Confidence score was obtained as 0.98. Fig. 4 (e) shows the precision/recall curve generated from the validation set after training completes with car: 0.969 mAP@0.5, three-wheel: 0.994 mAP@0.5, bus: 0.994 mAP@0.5, truck: 0.987 mAP@0.5, motorbike: 0.963 mAP@0.5 and van: 0.985 mAP@0.5 and all classes: 0.982 mAP@0.5. Fig. 4 (f) illustrates the regression losses for bounding box, object loss, and class loss, this includes training and validation and also the performance metrics for 300 epochs.

B. Hyperparameter Evolution

About 30 hyperparameters are employed in YOLOv5 for diverse training settings. It is crucial to initialize these variables correctly before evolving since better initial assumptions will result in better final results. Hyperparameter evolution was carried out on the best model generated from the initial training results. The model evolved for 300 generations for 10 epochs with a batch size of 32. The best generation was produced at 169th output. Table 1 shows the initial values and the best-evolved values for the hyperparameters.

Table. 1. Initial and evolved hyperparameters

Hyperparameter	Initial value	Evolved value
initial learning rate (SGD=1E-2, Adam=1E-3)	0.01	0.00554
final OneCycleLR learning rate (lr0 * lrf)	0.01	0.01
SGD momentum/Adam beta1	0.937	0.96721
optimizer weight decay 5e-4	0.0005	0.00043
warmup epochs (fractions ok)	3.0	3.0585
warmup initial momentum	0.8	0.8759
warmup initial bias lr	0.1	0.1342
box loss gain	0.05	0.03957
cls loss gain	0.5	0.47055
cls BCELoss positive_weight	1.0	1.0945
obj loss gain (scale with pixels)	1.0	0.94503
obj BCELoss positive_weight	1.0	0.91331
IoU training threshold	0.20	0.2
anchor-multiple threshold	4.0	4.7919
anchors per output layer (0 to ignore)	3	3.1902
focal loss gamma (efficientDet default gamma=1.5)	0.0	0.0
image HSV-Hue augmentation (fraction)	0.015	0.01278
image HSV-Saturation augmentation (fraction)	0.7	0.67957
image HSV-Value augmentation (fraction)	0.4	0.41589
image rotation (+/- deg)	0.0	0.0

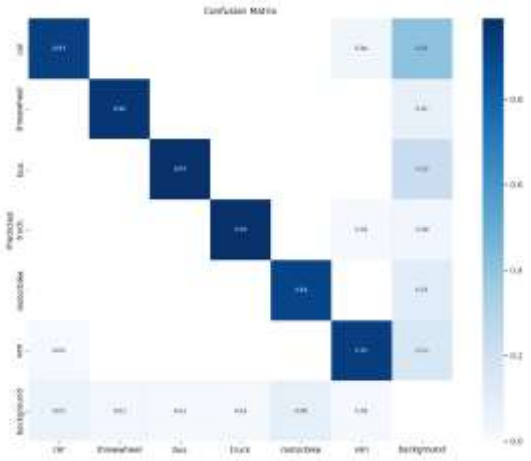

Fig.5(a)

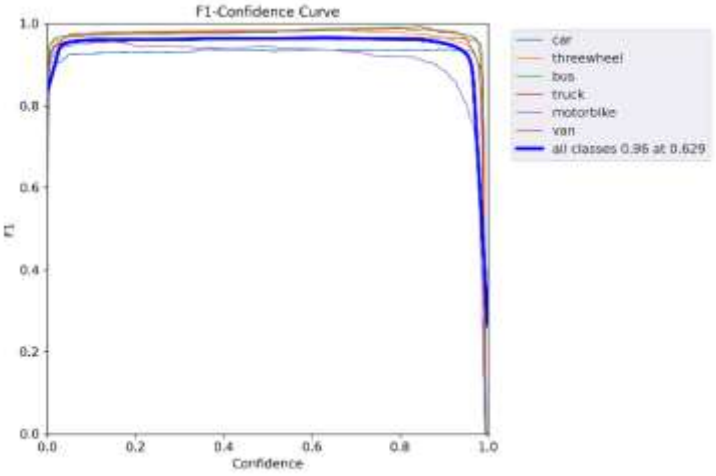

Fig.5(b)

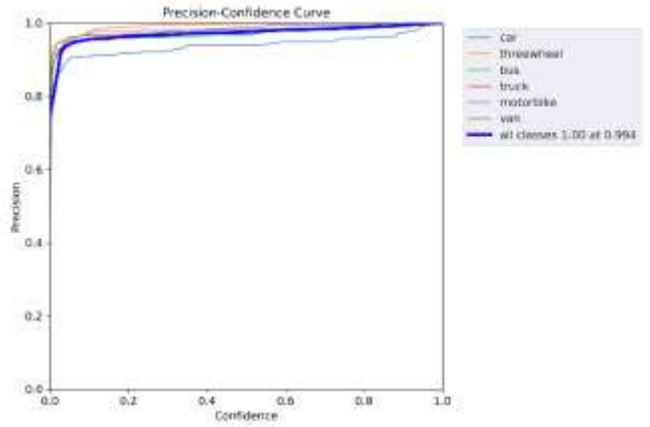

Fig.5(c)

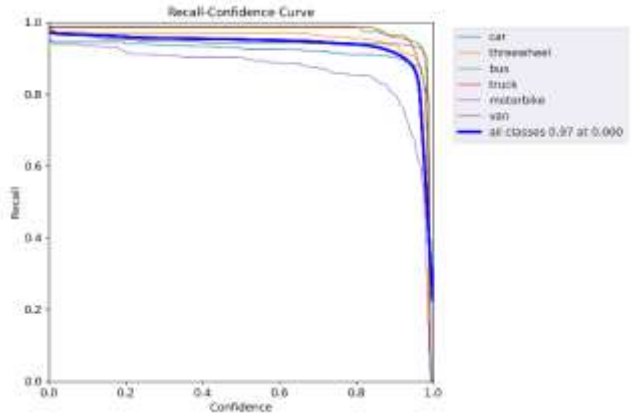

Fig.5(d)

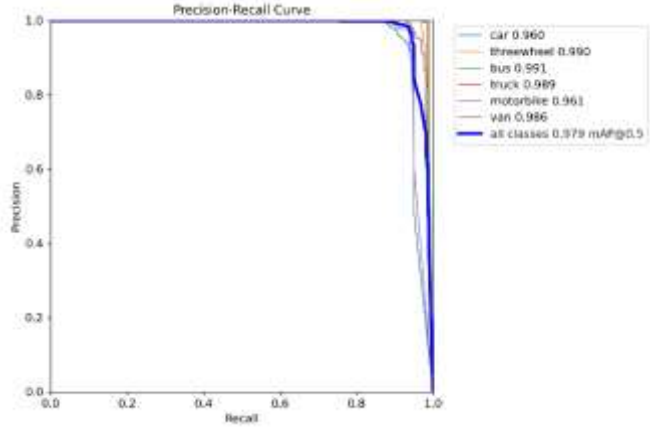

Fig.5(e)

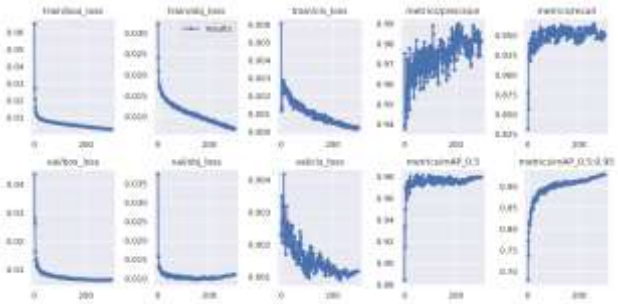

Fig.5(f)

Fig. 5. Training results with evolved hyperparameters

The class: car, bus, and truck had the same predictions as from the initial training settings. 96% of samples were predicted as TP for three-wheel, 91% for motorbikes, and 94% for vans whereas 1% of the truck and 4 % of the car were confused as

Table. 2. Comparison of model results with batch size, epochs, and hyperparameters

	Batch Size	Epochs	Precision (%)	Recall (%)	mAP		Accuracy (%)
					mAP@.5 (%)	mAP@.5:.95 (%)	
Default Hyperparameters	64	100	95.7	93.5	97.0	87.5	93.2
	32	260	97.5	95.3	98.2	92.1	96.0
Evolved Hyperparameters	64	300	97.5	93.8	97.7	91.6	94.6
	32	300	98.2	94.9	97.9	92.8	95.3

shown in Fig. 5 (a). Fig.5 (b) shows the F1-Confidence score was obtained as 0.96 at 0.629 confidence. Fig. 5 (c) presents the Precision-Confidence score obtained as 1.00 at 0.994 confidence. Fig. 5 (d) presents the Recall-Confidence score obtained as 0.97. Fig. 5 (e) shows the precision/recall curve generated from the validation set after training completes with car: 0.969 mAP@0.5, three-wheel: 0.990 mAP@0.5, bus: 0.991 mAP@0.5, truck: 0.989 mAP@0.5, motorbike: 0.961 mAP@0.5 and van: 0.986 mAP@0.5 and all classes: 0.979 mAP@0.5. Fig. 5 (f) illustrates the final results for regression losses and performance metrics from training with evolved hyperparameters.

Table. 3. Model results of each class

Class	Precision (%)	Recall (%)	mAP		Accuracy (%)
			mAP@.5 (%)	mAP@.5:.95 (%)	
car	94.9	92.4	96.0	94.0	93.0
three-wheel	99.9	96.9	99.0	94.6	96.0
bus	98.5	98.9	99.1	96.7	99.0
truck	97.9	98.7	98.9	93.8	99.0
motorbike	99.5	88.6	96.1	81.2	91.0
van	98.5	94.1	98.6	96.5	94.0
all	98.2	94.9	97.9	92.8	95.3

The model was trained for batch sizes 32 and 64 for default and evolved hyperparameters. The number of epochs was determined during training to monitor if the model overfits. An accuracy of 93.2% was obtained for batch size 64 trained for 100 epochs, following the evolved hyperparameters trained with the same batch size for 300 epochs, an accuracy of

94.6% was obtained. Higher batch sizes usually do not achieve higher accuracies; therefore, the batch size was reduced to 32. An accuracy of 96% was obtained for 260 epochs with default hyperparameters. Final model accuracy was obtained as 95.3% where batch size: 32 and epochs: 300 trained with the evolved hyperparameters.

The final model results: precision, recall, mAP, and accuracies of each class are shown in Table. 3. The overall precision of the model was obtained as 98.2%, the recall was obtained as 94.9%, mAP@.5 was 97.9%, mAP@.5:.95 was 92.8% and the model achieved overall recognition accuracy of 95.3%. The following experiment has shown satisfactory results, considering the knowledge gaps identified in the existing research.

4 Conclusions

This paper proposes a model for categorizing vehicle types utilizing the most recent state-of-the-art object detection model, YOLOv5. With an overall mAP@.5 of 97.9% and accuracy of 95.3%, the model's promising results. The model was very successful in recognizing the said vehicles on the road, as shown by the results. Any future transportation system that has to accurately identify the type of vehicle can easily incorporate it. While the classes can be further broken down into a more detailed manner, dividing classes of vehicles as SUVs, sedans, crossovers, jeeps, etc., Following the experiment, it was clear, that the model produced better results concerning the precision and recall, and it is determined that the model could achieve best results, greater quality photos must be utilized to train the model to allow it to extract more features from the data

References:

[1] Lin, M. Zhao, X. (2019) 'Application Research of Neural Network in Vehicle Target Recognition and Classification', 2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), 5–8. doi: 10.1109/ICITBS.2019.00010.

[2] Kim, J.-A., Sung, J.-Y. και Park, S.-H. (2020) 'Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition', 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), 1–4. doi: 10.1109/ICCE-Asia49877.2020.9277040.

[3] Yang, Y. (2020) 'Realization of Vehicle Classification System Based on Deep Learning', 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), 308–311. doi: 10.1109/ICPICS50287.2020.9202376.

[4] Aishwarya, C. N., Mukherjee, R. Mahato, D. K. (2018) 'Multilayer vehicle classification integrated with single frame optimized object detection framework using CNN based deep learning architecture', 2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT). 1–6. doi: 10.1109/CONECCT.2018.8482366.

[5] Htet, K. S. Sein, M. M. (2020) 'Event Analysis for Vehicle Classification using Fast RCNN', 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE), 403–404. doi: 10.1109/GCCE50665.2020.9291978.

[6] MWP Maduranga, Dilshan Nandasena, "Mobile-Based Skin Disease Diagnosis System Using Convolutional Neural Networks (CNN)", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.14, No.3, pp. 47-57, 2022.DOI: 10.5815/ijigsp.2022.03.05

[7] Doan, T.-N. Truong, M.-T. (2020) 'Real-time vehicle detection and counting based on YOLO and DeepSORT', 2020 12th International Conference on Knowledge and Systems Engineering (KSE), 67–72. doi: 10.1109/KSE50997.2020.9287483.

[8] Roecker, M. N. (2018) 'Automatic Vehicle type Classification with Convolutional Neural Networks', 2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP), 1–5. doi: 10.1109/IWSSIP.2018.8439406.

[9] Harianto, R. A., Pranoto, Y. M. Gunawan, T. P. (2021) 'Data Augmentation and Faster RCNN Improve Vehicle Detection and Recognition', 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT), 128–133. doi: 10.1109/EIConCIT50028.2021.9431863.

[10] Maungmai, W. Nuthong, C. (2019) 'Vehicle Classification with Deep Learning', 2019 IEEE 4th International Conference on Computer and Communication Systems

(ICCCS), 294–298. doi: 10.1109/CCOMS.2019.8821689.

[11] Wang, X. (2019) 'Real-time vehicle type classification with deep convolutional neural networks, Journal of Real-Time Image Processing, 16(1), 5–14. doi: 10.1007/s11554-017-0712-5.

[12] San, W. J., Lim, M. G. Chuah, J. H. (2018) 'Efficient Vehicle Recognition and Classification using Convolutional Neural Network', 2018 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), 117–122. doi: 10.1109/I2CACIS.2018.8603700.

[13] Redmon, J. (2015) 'You Only Look Once: Unified, Real-Time Object Detection', CoRR, abs/1506.02640. http://arxiv.org/abs/1506.02640.

[14] W. T. S. Wickramanayake, P. K. T. V. Kumararathhe, U. G. B. A. Jayashan and M. Maduranga, "ParkMate: An Image Processing Based Automated Car Parking System," 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT), 2022, pp. 1-5, doi: 10.1109/IC3IOT53935.2022.9767991.

[15] Redmon, J. Farhadi, A. (2018) 'YOLOv3: An Incremental Improvement', CoRR, abs/1804.02767. http://arxiv.org/abs/1804.02767.

[16] Bochkovskiy, A., Wang, C.-Y. και Liao, H.-Y. M. (2020) 'YOLOv4: Optimal Speed and Accuracy of Object Detection', CoRR, abs/2004.10934. https://arxiv.org/abs/2004.10934.

[17] Chen, Y. (2017) 'Vehicle type classification based on convolutional neural network', 2017 Chinese Automation Congress (CAC), 1898–1901. doi: 10.1109/CAC.2017.8243078.

[18] Armin, E. U., Bejo, A. Hidayat, R. (2020) 'Vehicle Type Classification in Surveillance Image based on Deep Learning Method', 2020 3rd International Conference on Information and Communications Technology (ICOIACT), 400–404. doi: 10.1109/ICOIACT50329.2020.9332047.

[19] Jocher, G., 2020. GitHub - ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite. [online] GitHub. Available at: <https://github.com/ultralytics/yolov5> [Accessed 23 June 2022].

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

Nadin Pethiyagoda, carried out the dataset preparation, model training and evaluation, MWP Maduranga, methodology and technical writing. DMR Kulasekara and TL Weerawardena overall supervision of the work.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself
No funding was received for conducting this study.

Conflict of Interest
The authors have no conflicts of interest to declare that are relevant to the content of this article.