Enhancing Public Transportation Detection using YOLOv5

Nur Liyana Ameera Arbi Shukhair
Electrical and Computer Engineering Department
International Islamic University Malaysia
53100 Kuala Lumpur, Malaysia
nurameera345@gmail.com

Teddy Surya Gunawan
Electrical and Computer Engineering Department
International Islamic University Malaysia
53100 Kuala Lumpur, Malaysia
tsgunawan@iium.edu.my

Hasmah Mansor
Electrical and Computer Engineering Department
International Islamic University Malaysia
53100 Kuala Lumpur, Malaysia
hasmahm@iium.edu.my

Kushsairy Abdul Kadir
Electrical Section
Universiti Kuala Lumpur-British Malaysian Institute
53100 Kuala Lumpur, Malaysia
kushsairy@unikl.edu.my

***Abstract* — In the current context of urbanization and transportation expansion, the need for accurate and efficient detection systems for public transportation has become of the utmost importance. The paper presents a novel strategy to establish new standards in transportation detection systems. Using the power of the YOLOv5 deep learning algorithm, the dataset is divided into training, testing, and validation segments to ensure a thorough evaluation. With a training dataset size of 75% and a test-validation split of 25%, our methodology showcases a compelling mean Average Precision (mAP) value of 0.973. Our findings highlight a precision of 0.971, pointing to accurate predictions in approximately 97.1% of cases, and a recall of 0.953, underscoring the model's efficiency in capturing around 95.3% of relevant objects. Such results, particularly the distinguishable taxi class among similar objects, represent significant improvements over previous benchmarks. The model's prowess is evident in its ability to distinguish even in situations involving entities that closely resemble one another, such as taxis and police cars. Our proposed system excels with increased accuracy, precision, and F1 scores compared to a standard study. This paper concludes that with the strategic application of YOLOv5, the future of public transportation detection systems is bright and on the cusp of a new era of efficiency and precision.**

Keywords—public transportation, object detection, computer vision, deep learning, YOLOv5.

I. INTRODUCTION

The public transportation system, a cornerstone of modern urban mobility, is perpetually challenged by insufficient real-time detection capabilities. Historically, vehicle detection systems were developed with the primary objective of pinpointing vehicles in real-time, with applications ranging from traffic management and toll collection to broader applications in the realm of public transportation [1, 2]. It is evident that efficient public transportation, essential for locals and tourists, provides a balance of convenience and economy. Nevertheless, the urban landscape of today, marred by rampant congestion, safety loopholes, and a glaring lack of real-time data, requires immediate improvements [3].

The importance of intelligent vehicle detection and classification cannot be overstated in an era of rapid urbanization and increasing traffic congestion. Advanced machine learning algorithms have emerged as pivotal tools in this endeavor, catalyzing transformation in highway and traffic management domains. Due to their varied sizes and shapes, vehicles present a unique challenge regarding detection. This barrier must be surmounted to implement effective traffic management strategies [4]. Nevertheless, the ramifications of robust vehicle detection go far beyond mere identification. By meticulously classifying these vehicular entities, traffic management systems can collect invaluable information regarding traffic flow dynamics, congestion levels, and the sheer volume of vehicles [5]. Armed with this granular data, traffic management authorities can calibrate traffic signal timings, assign lanes, and implement dynamic traffic control measures. Such interventions improve the effectiveness of highways and pave the way for more refined traffic control mechanisms, resulting in safer and more efficient transportation ecosystems [6].

Historically, background updates [7], optical flow [8], and continuous video frame difference [9] were the mainstays of object detection, particularly in public transportation systems. Although these methodologies were ground-breaking at the time of their inception, they have obvious limitations when applied to complex and continuously dynamic environments. A paradigm shift in detection methods is perceptible, with a notable trend toward deep learning-based tactics. Convolutional neural networks (CNNs), exemplified by algorithms such as the You Only Look Once (YOLO) series, illustrate this evolution by demonstrating outstanding performance benchmarks and unrivaled robustness in detecting and classifying objects in transportation scenarios [10, 11].

The primary objective of this research is to design a system for the precise detection and classification of diverse public vehicles. Utilizing the cutting-edge YOLOv5 algorithm, renowned for its superior detection capabilities, this study aims to improve the precision and dependability of transportation services significantly.

II. PUBLIC TRANSPORT DETECTION USING VARIOUS YOLO ALGORITHMS

In recent years, many research efforts have been devoted to evaluating the effectiveness of various object detection algorithms, mainly as applied to public transportation detection and classification systems. Understanding the adaptability of the YOLO algorithms, particularly versions YOLOv3, YOLOv4, and YOLOv5, to the unique challenges posed by public transportation scenarios has been a significant focus of these studies. Such obstacles include various vehicle shapes, complex backgrounds, and the inherent dynamism of public transportation environments.

A. YOLOv3

In [12], YOLOv3 demonstrated superior performance to Faster R-CNN in vehicle detection, achieving a 99.07 percent accuracy rate compared to 79.40 percent for Faster R-CNN. The central focus of this study was to determine the optimal algorithm for identifying each vehicle instance in an image. The results indicated that YOLOv3 was superior to Faster R-CNN at recognizing all car instances within an image. However, YOLOv3 encountered difficulties when detecting objects of medium to large size. It prompted the development of YOLOv4, proposed in 2020, to address this specific deficiency.

B. *YOLOv4*

The potential benefits of optimizing GPU resources to augment YOLOv4's performance and bolster GPU-CPU interactivity were reported in [13]. Their results suggested that YOLOv4, besides being faster, also demonstrated enhanced accuracy over YOLOv3. A significant advantage presented by YOLOv4 is its simplicity and straightforwardness in implementation. It is proficient enough to operate seamlessly on a standard GPU, catering to inference and training phases.

C. *YOLOv5*

Emerging as an advancement over YOLOv4, YOLOv5 was proposed to be leaner and swifter without compromising accuracy. It was materialized by embracing state-of-the-art techniques such as a single-stage architecture, PANet, and the integration of EfficientNet-Lite backbones. Evaluative studies on the COCO dataset revealed that YOLOv5 exemplified commendable precision, with an mAP approaching 50 and ensuring swift processing. Further, a study in [11] indicated that YOLOv5 achieved a mAP50 score of 0.8302, outpacing other models like VGG16 and VGG19. Such findings underscore YOLOv5's superiority in terms of accuracy for specific tasks or datasets. Overall, YOLOv5 emerges as a potent object detection tool, offering superior speed, precision, and adaptability in contrast to its YOLO predecessors. Like YOLOv4, it can also efficiently operate on a standard GPU for inference and training.

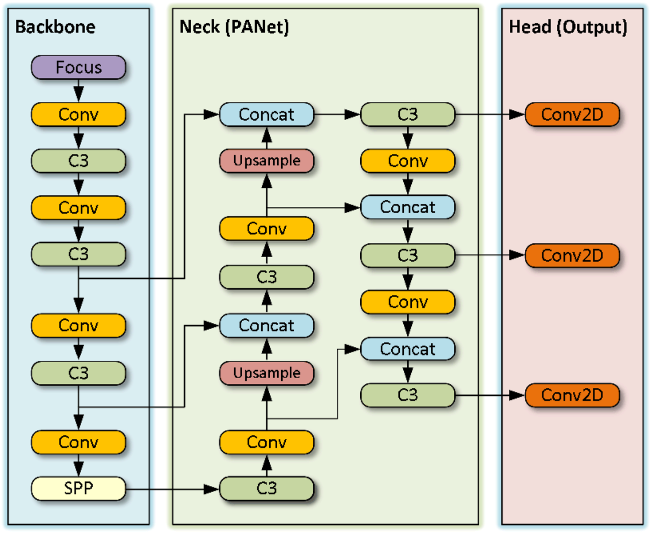


Fig. 1. YOLOv5 Architecture (adopted from [14])

Given the various advancements and enhancements across YOLO versions, our research favored YOLOv5 for multiple reasons. Among them, the model's architectural efficiency is the most prominent. YOLOv5 includes a streamlined single-stage architecture, PANet integration, and EfficientNet-Lite backbones, as described in the preceding sections. These components not only facilitate a reduction in computational overhead but also endow the model with the capacity to process information quickly without sacrificing precision.

YOLOv5 consistently outperforms its predecessors and other contemporary models in specific tasks or datasets, such as VGG16 and VGG19. Given that our research hinges on high precision and efficiency in public transport detection scenarios, employing an algorithm that balances speed, precision, and flexibility is essential. With its demonstrated capabilities and empirical successes, YOLOv5 perfectly aligns with our research objectives and the domain's challenges. Therefore, YOLOv5 was the logical and strategic choice for our investigations.

III. Design and Implementation

This section discussed the design and implementation of our proposed algorithm, including system requirements, dataset collection, image preprocessing, model training, and testing.

A. *System Requirements*

System requirements form the basis of any project, ensuring its viability and efficiency. For this study, it has been divided into hardware and software. For the hardware part, we utilized the computational advantages of Google Colab, specifically its CPU runtime, which an Intel Xeon CPU characterizes clocked at 2.20 GHz, 13 GB RAM, a Tesla K80 accelerator, and 12 GB GDDR5 VRAM. The free Google Colab T4 GPU was a crucial component, especially for the efficient training of models. This robust configuration enabled rapid data processing and maintained consistent performance throughout the rigorous phases of model training.

For the software part, Roboflow was our primary instrument for image preprocessing and data cleansing. Google Colab utilized Python (version 3.9.13), renowned for its ease of use and abundance of machine learning libraries, via the Jupyter Notebook. PyTorch and the essential libraries in Table I were selected as our framework.

TABLE I. Libraries and Modules Used in Research

No	Open libraries/modules	Version
1	Pytorch	1.13.1
2	Anaconda	22.9.0
3	Sklearn (scikit-learn)	1.0
4	Numpy	1.17.3
5	Matplotlib	3.1.3
6	OpenCV	4.7.0

B. *Dataset Collection*

Given its centrality to model accuracy, collecting an adequate dataset was our top priority. By scouring platforms such as Kaggle and Roboflow Universe, we acquired a dataset containing a variety of public vehicles, including buses, taxis, and trams. This dataset was preprocessed in preparation for model training. This dataset was preprocessed in preparation for model training. The dataset is diverse in terms of the types of buses, taxis, and trains represented, as well as the backgrounds in which the images were taken. However, the dataset may contain biases since it was collected from a variety of online sources. For example, the dataset may be overrepresented with images of certain types of buses, taxis, and trains. To mitigate this bias, we filtered the dataset to ensure that each type was represented equally. Table II summarizes the key characteristic of the datasets.

TABLE II. DATASET CHARACTERISTICS

Characteristic	Description
Number of samples	4177 images
Number of classes	3 (bus, taxis, trains)
Diversity	Diverse in terms of types, backgrounds, etc
Potential biases	May be overrepresented with images of certain types
Mitigation steps	Filtered the dataset to ensure equal representation of all types

C. Image Preprocessing

Utilizing Roboflow, we refined images of buses, taxis, and tram-trains through a preprocessing pipeline that involved cropping and resizing to emphasize unique features. The dataset was partitioned strategically, allocating 75% for training and the remaining 25% for validation and testing. This preprocessing not only improved the data quality but also significantly enhanced the performance and accuracy of the machine learning model. Figure 2 shows the preprocessed image examples.

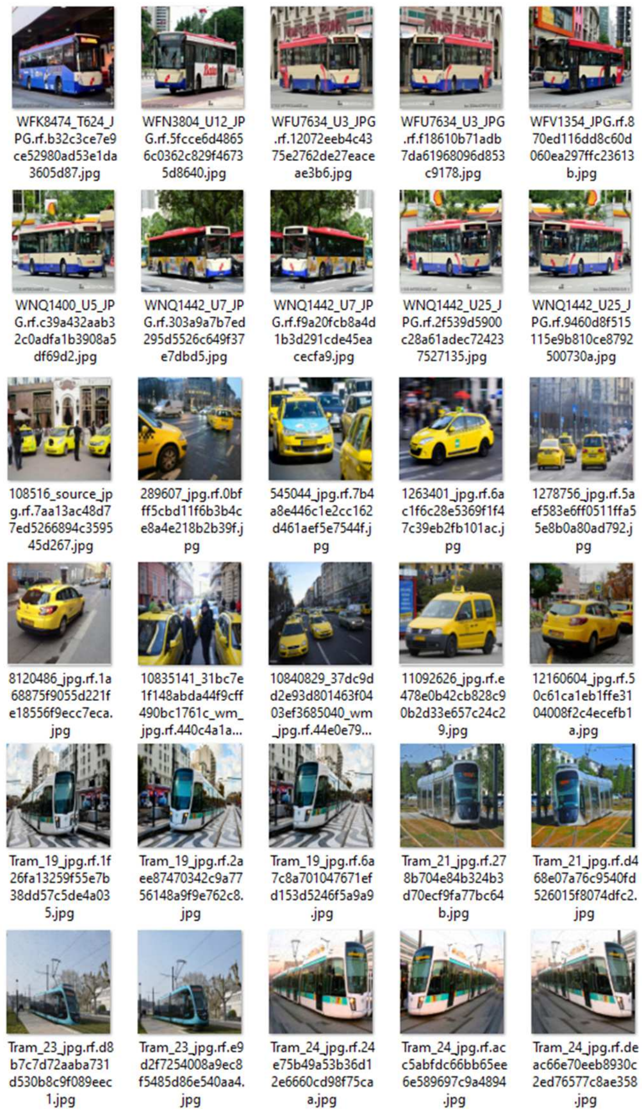

WFK8474_T624_JPG.rf.b32c3ce7e9ce52980ad53e1da3605d87.jpg
WFN3804_U12_JPG.rf.5fcce6d48656c0362c829f46735d8640.jpg
WFU7634_U3_JPG.rf.12072eeb4c4375e2762de27eaceae3b6.jpg
WFU7634_U3_JPG.rf.f18610b71adb7da61968096d853c9178.jpg
WFV1354_JPG.rf.870ed116dd8c60d060ea297ffc23613b.jpg
WNQ1400_U5_JPG.rf.c39a432aab32c0adfa1b3908a5df69d2.jpg
WNQ1442_U7_JPG.rf.303a9a7b7ed295d5526c649f37e7dbd5.jpg
WNQ1442_U7_JPG.rf.f9a20fcb8a4d1b3d291cde45eacecfa9.jpg
WNQ1442_U25_JPG.rf.2f539d5900c28a61adec724237527135.jpg
WNQ1442_U25_JPG.rf.9460d8f515115e9b810ce8792500730a.jpg
108516_source_jpg.rf.7aa13ac48d77ed5266894c359545d267.jpg
289607_jpg.rf.0bfff5cbd11f6b3b4ce8a4e218b2b39f.jpg
545044_jpg.rf.7b4a8e446c1e2cc162d461aef5e7544f.jpg
1263401_jpg.rf.6ac1f6c28e5369f1f47c39eb2fb101ac.jpg
1278756_jpg.rf.5aef583e6ff0511ffa55e8b0a80ad792.jpg
8120486_jpg.rf.1a68875f9055d221fe18556f9ecc7eca.jpg
10835141_31bc7e1f148abda44f9cff490bc1761c_wm_jpg.rf.440c4a1a...
10840829_37dc9dd2e93d801463f0403ef3685040_wm_jpg.rf.44e0e79...
11092626_jpg.rf.e478e0b42cb828c90b2d33e657c24c29.jpg
12160604_jpg.rf.50c61ca1eb1ffe3104008f2c4ecefb1a.jpg
Tram_19_jpg.rf.1f26fa13259f55e7b38dd57c5de4a035.jpg
Tram_19_jpg.rf.2aee87470342c9a7756148a9f9e762c8.jpg
Tram_19_jpg.rf.6a7c8a701047671efd153d5246f5a9a9.jpg
Tram_21_jpg.rf.278b704e84b324b3d70ecf9fa77bc64b.jpg
Tram_21_jpg.rf.d468e07a76c9540fd526015f8074dfc2.jpg
Tram_23_jpg.rf.d8b7c7d72aaba731d530b8c9f089eec1.jpg
Tram_23_jpg.rf.e9d2f7254008a9ec8f5485d86e540aa4.jpg
Tram_24_jpg.rf.24e75b49a53b36d12e6660cd98f75caa.jpg
Tram_24_jpg.rf.acc5abfdc66bb65ee6e589697c9a4894.jpg
Tram_24_jpg.rf.deac66e70eeb8930c2ed76577c8ae358.jpg

Fig. 2. Examples of preprocessed images for buses, taxis, and tram-trains

The streamlined steps in Roboflow were:

1. *Data Upload*: Images were uploaded to Roboflow via its API.
2. *Image Annotation*: Objects in images were labeled with bounding boxes and class labels, vital for precise object detection and categorization.
3. *Data Augmentation*: The dataset was enhanced with techniques like rotation and cropping.
4. *Data Cleaning*: Low-quality or wrongly labeled images were filtered to ensure reliability.
5. *Data Normalization and Split*: Post-normalization, the data was partitioned into training, testing, and validation subsets.
6. *Data Formatting*: The dataset was structured for PyTorch, our chosen machine learning framework.
7. *Exportation*: The dataset expanded from 3,068 to 4,177 images post preprocessing.

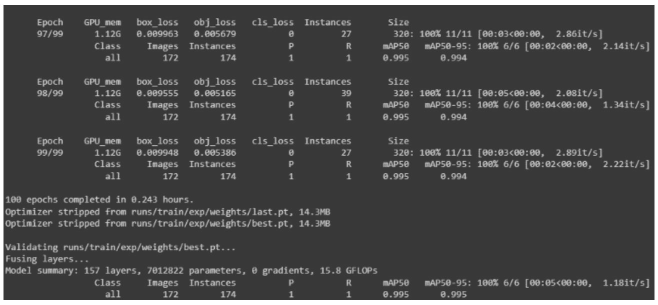
D. Model Training

Several preparatory steps were required before model training could begin. Since our chosen algorithm, YOLOv5, is dependent on PyTorch, we started by installing all the necessary dependencies, focusing first on PyTorch. We decided to install PyTorch via Anaconda due to the Windows operating system. After downloading and installing the most recent version of Anaconda (22.9.00), we retrieved the PyTorch installation command from its official website and ran it from the command prompt. This ensured all necessary packages were installed and compatibility with Python was verified.

We shifted our focus to YOLOv5. We cloned the Ultralytics repository for YOLOv5 and installed it in a Google Colab notebook. After successful cloning, we accessed the necessary files to implement the YOLOv5 algorithm. Installing YOLOv5 dependencies and importing the required libraries and packages into the Google Colab environment were the final steps in the setup process. The dataset was then loaded using the **data.yaml** file, which pointed to the training and validation image directories. The actual model training required only a single line of code. The model was trained for 100 epochs with a batch size of 16, while the number of workers was restricted to two to prevent system overload.

E. Model Evaluation

During training, the weights of the YOLOv5 model were saved in a subfolder named "weights" within the YOLOv5 directory. This process created two crucial files: **best.pt** and **last.pt**. The **best.pt** file contains the optimal weights determined by the model's performance throughout the training process, while the **last.pt** file stores the model's weights after the final epoch. These weight files can be used for various purposes, including retraining and deploying the model for predictions.



```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      97/99      1.12G   0.009963   0.005679          0         27        320: 100% 11/11 [00:03<00:00,  2.86it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:02<00:00,  2.14it/s]
                   all        172        174          1          1      0.995      0.994

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      98/99      1.12G   0.009555   0.005165          0         39        320: 100% 11/11 [00:05<00:00,  2.08it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:04<00:00,  1.34it/s]
                   all        172        174          1          1      0.995      0.994

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      99/99      1.12G   0.009948   0.005386          0         27        320: 100% 11/11 [00:03<00:00,  2.89it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:02<00:00,  2.22it/s]
                   all        172        174          1          1      0.995      0.994

100 epochs completed in 0.243 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.3MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:05<00:00,  1.18it/s]
                   all        172        174          1          1      0.995      0.995
```

Fig. 3. Example of mAP preliminary results

The model underwent an evaluation phase after training. Standard in object detection tasks, mean Average Precision (mAP), was used for this evaluation, as shown in Figure 2. mAP evaluates the model's ability to identify objects within images. It assigns a score between 0 and 1. Greater mAP values indicate enhanced object detection performance. This mAP evaluation was conducted in the Google Colab environment following the project's emphasis on precision.

IV. Results and Discussion

In this section, we evaluate the model's performance across multiple metrics, including precision, recall, and training loss, as well as its ability to generalize to new data. In addition to comparing the proposed work and benchmark models, this section concludes with a summary of key insights.

A. Model Training Evaluation

During the training and evaluation phase, the YOLOv5 algorithm was used for object detection and classification tasks on Google Colab. The number of training epochs was increased from 100 to 150 due to a strategic adjustment based on performance metrics intended to improve model accuracy. Subsequent analysis revealed a positive correlation between the number of epochs and training precision, validating the adjustment's effectiveness. The model's impressive mean Average Precision (mAP) of 0.973% indicates exceptional proficiency in detecting and classifying diverse forms of public transportation. Additional analysis revealed a precision rate of 0.971, indicating that approximately 97.1 percent of the model's classifications were accurate, thereby minimizing false positives. In addition, a recall value of 0.953 was attained, demonstrating the model's ability to correctly identify approximately 95.3% of relevant objects within the dataset, thereby reducing false negatives. These metrics show the robustness of the model in detecting and classifying public transportation.

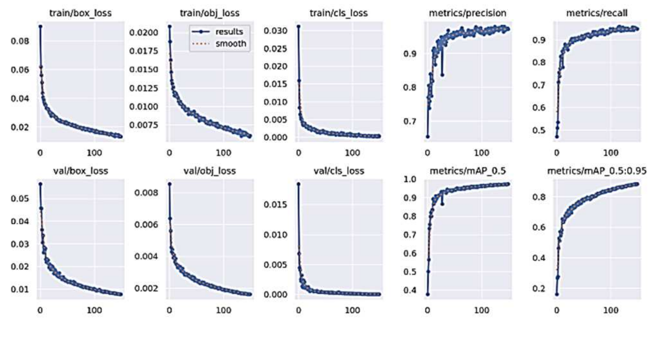


Fig. 4. Model Training Evaluation Using Various Metrics

In Figure 4, we scrutinize the progression of critical metrics—box loss, objective loss, and class loss—throughout the training and validation phases. These metrics offer valuable insights into the model's performance in three critical areas: accurate object localization (box loss), discriminative power (objective loss), and categorical classification (class loss). For instance, reducing box loss over time would indicate improving proficiency in pinpointing objects within an image. The figure also presents mAP0.5, which stands for mean Average Precision at a 0.5 threshold, as a baseline measure of the model's predictive accuracy. To provide a more nuanced evaluation, mAP0.5:0.95 extends the threshold to 0.95. This higher threshold requires a stringent evaluation criterion, revealing the model's ability to maintain accuracy under more demanding conditions. Analyzing these metrics in tandem offers a well-rounded perspective on the model's overall effectiveness and potential areas for optimization.

Constructed with an Intersection over Union (IoU) threshold of 0.5, the confusion matrix provides crucial insights into the model's classification abilities across various transport categories. While the model excels at identifying "bus" and "tram-train" categories, it has difficulty identifying "taxi" vehicles. A true positive rate of 0.92 for "taxi" indicates that the model correctly identifies 92 percent of the taxis in the dataset. Still, the large number of false negatives suggests that there is significant room for improvement. In particular, an 8 percent false negative rate means that the model incorrectly classifies these instances as "background," indicating its limited ability to distinguish subtler characteristics that distinguish taxis from the environment. A second false negative value of 0.90 is also concerning, as it indicates that the model incorrectly identifies "background" class elements as "taxi." While these obstacles do not undermine the model's overall efficacy, they call attention to potential areas for improvement. Absolute accuracy in deep learning is an elusive goal, but the model's performance in the "taxi" category leaves room for improvement.

A comparative analysis is conducted to assess the significance and efficacy of the proposed work compared to [15]. This benchmark was selected due to its emphasis on vehicle detection and classification, providing pertinent context for our research. Both the benchmark and the proposed work employ the same algorithms and performance metrics, allowing for a fair and balanced evaluation. This comparison provides a holistic view of the proposed work's strengths, limitations, and advancements, as presented in Table III.

TABLE III. Model Training Comparison

Metric	[15]	Our Work
Image input size	640 x 640	640 x 640
Dataset size	2400 images	4177 images
Precision	0.83	0.971
F1 score	0.79	0.953
mAP0.5 (accuracy)	0.8302	0.973

Table III provides a nuanced comparison between our model and the benchmark study [15], highlighting that while both utilize a 640 x 640 pixel image input size, our model benefits from a larger dataset of 4,177 images instead of 2,400 for the benchmark. This expanded dataset correlates with superior precision, F1 score, and mAP0.5 performance. However, despite being encouraging, these metrics raise essential questions. The increased precision of 0.971% may indicate possible overfitting due to the larger data set. In addition, the model's elevated F1 score of 0.953 suggests improvements in both precision and recall. Still, it does not account for its specific deficiencies, such as difficulties with false negatives in the "taxi" category. In addition, the impressive mAP0.5 of 0.973% is a strong indicator of model accuracy.

B. Model Testing Evaluation

This section evaluates the trained model's object detection and classification performance, focusing solely on static images. We aim to assess the model's generalization capabilities and effectiveness in real-world scenarios by applying it to previously unseen images.

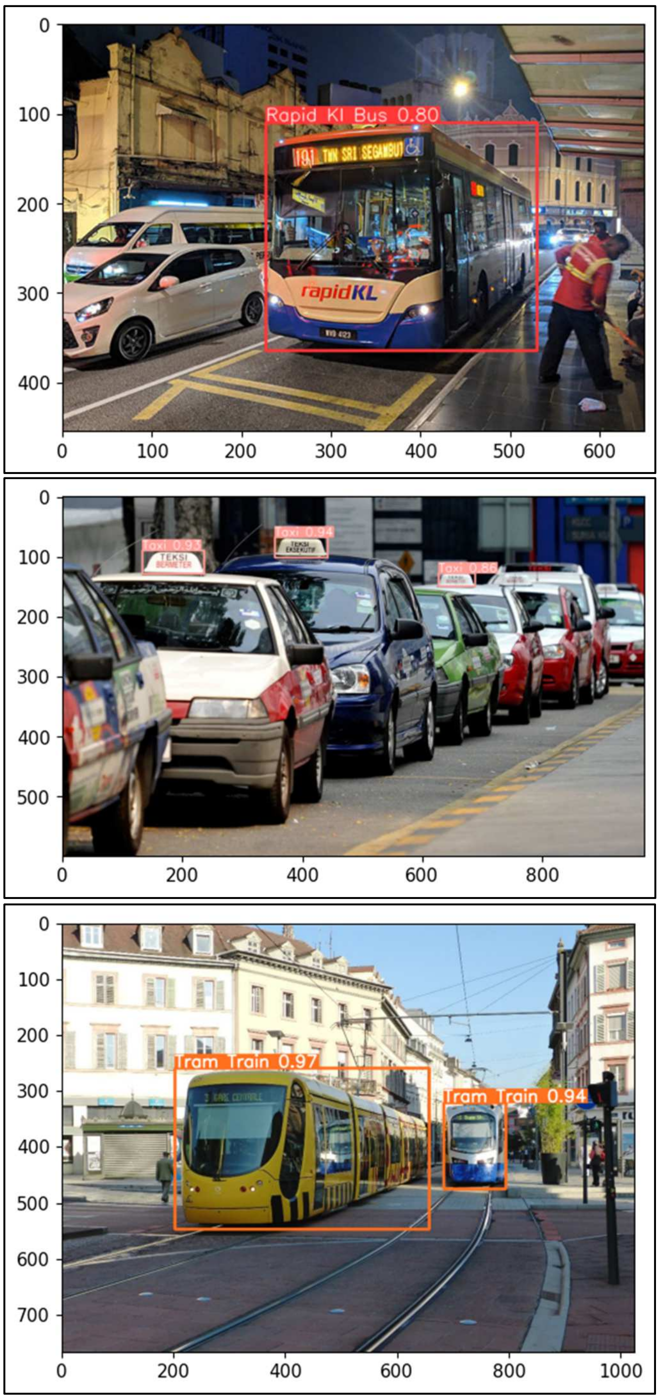


Fig. 5. Test results for bus, taxi, and tram-train

Figure 5 illustrates the model's performance on static image inputs by generating color-coded bounding boxes representing different detected object classes (red for 'Bus,' pink for 'Taxi,' and orange for 'Tram Train'). While this color-coding improves interpretability, it also raises serious concerns about the accuracy and dependability of the model. In applications such as automated traffic monitoring or autonomous driving, any misalignment between the color of the bounding box and the actual object class—such as an orange box incorrectly identifying a 'Bus' as a 'Tram Train'—would indicate a classification error with real-world consequences. In addition, it is essential to evaluate the robustness of these color classifications under varying lighting conditions, object occlusions, and other complicating environmental factors. In addition, the model's efficacy in cluttered environments with multiple classes in close proximity must be evaluated, as performance degradation in such situations could lead to misclassifications. This multifaceted analysis identifies the strengths and limitations of the model's object detection capabilities.

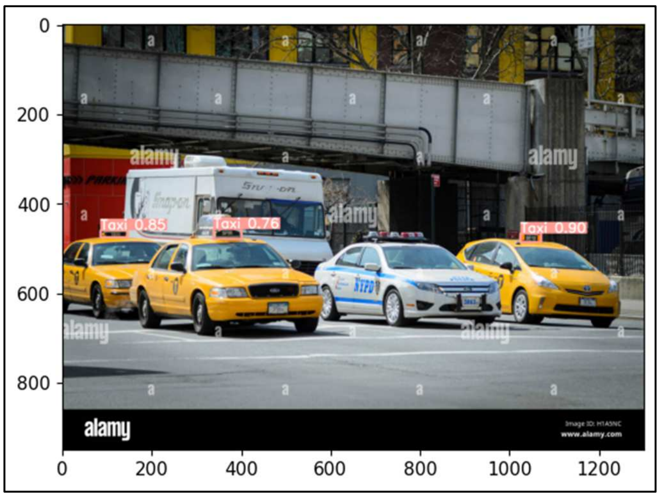


Fig. 6. Detection of Taxi with Police Car nearby

Figure 6 demonstrates the model's precise ability to identify and differentiate between closely related objects, such as a police car. In the given scenario, it successfully generates a bounding box around the 'Taxi' object while ignoring a police car of comparable size and appearance. It suggests that the model's feature-learning mechanism has high discrimination and robustness. However, the precision demonstrated could vary across diverse datasets and environmental conditions. Consequently, although the model's ability to differentiate between a 'Taxi' and a police car is promising, it would benefit from additional testing to confirm its consistency and generalizability, particularly in complex real-world scenarios.

V. Conclusions and Future Works

This paper successfully develops a detection system that can identify and classify three prevalent forms of public transportation: buses, taxis, and tram trains. Using static images and advanced machine learning algorithms, particularly YOLOv5, the system achieves high accuracy in vehicle classification, achieving an overall training accuracy of 97.3%. Multiple performance metrics, such as the Precision-Recall curve and the confusion matrix, provide additional evidence of the model's effectiveness. The system's accuracy, precision, recall, and F1 score superiority are also confirmed by comparison with benchmark works. For future works, there is potential to explore other YOLOv5 architectures and implement real-time detection using live video feeds, offering real-time insights for traffic management.

Acknowledgment

The authors would like to acknowledge the support from the International Islamic University Malaysia and UniKL-BMI for the research funding and facilities.

References

[1] S. Kul, S. Eken, and A. Sayar, "Distributed and collaborative real-time vehicle detection and classification over the video streams," *International Journal of Advanced Robotic Systems,* vol. 14, no. 4, p. 1729881417720782, 2017.

[2] Q. Cao, Z. Zhao, Q. Zeng, Z. Wang, and K. Long, "Real-time vehicle trajectory prediction for traffic conflict detection at unsignalized intersections," *Journal of advanced transportation,* vol. 2021, pp. 1-15, 2021.

[3] N. Serok, S. Havlin, and E. Blumenfeld Lieberthal, "Identification, cost evaluation, and prioritization of urban traffic congestions and their origin," *Scientific Reports,* vol. 12, no. 1, p. 13026, 2022.

[4] H. Song, H. Liang, H. Li, Z. Dai, and X. Yun, "Vision-based vehicle detection and counting system using deep learning in highway scenes," *European Transport Research Review,* vol. 11, no. 1, pp. 1-16, 2019.

[5] S. Cepni, M. E. Atik, and Z. Duran, "Vehicle detection using different deep learning algorithms from image sequence," *Baltic Journal of Modern Computing,* vol. 8, no. 2, pp. 347-358, 2020.

[6] K. Zaatouri and T. Ezzedine, "A self-adaptive traffic light control system based on YOLO," in *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, 2018: IEEE, pp. 16-19.

[7] H. Yang and S. Qu, "Real - time vehicle detection and counting in complex traffic scenes using background subtraction model with low - rank decomposition," *IET Intelligent Transport Systems,* vol. 12, no. 1, pp. 75-85, 2018.

[8] A. Gomaa, M. M. Abdelwahab, M. Abo-Zahhad, T. Minematsu, and R.-i. Taniguchi, "Robust vehicle detection and counting algorithm employing a convolution neural network and optical flow," *Sensors,* vol. 19, no. 20, p. 4588, 2019.

[9] J. Hu, R. Liu, Z. Chen, D. Wang, Y. Zhang, and B. Xie, "Octave convolution-based vehicle detection using frame-difference as network input," *The Visual Computer,* vol. 39, no. 4, pp. 1503-1515, 2023.

[10] J. Zhao *et al.*, "Improved vision-based vehicle detection and classification by optimized YOLOv4," *IEEE Access,* vol. 10, pp. 8590-8603, 2022.

[11] Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, and X. Guo, "Real-time vehicle detection based on improved yolo v5," *Sustainability,* vol. 14, no. 19, p. 12274, 2022.

[12] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3," in *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, 2019: IEEE, pp. 1-6.

[13] C.-J. Lin and J.-Y. Jhang, "Intelligent traffic-monitoring system based on YOLO and convolutional fuzzy neural networks," *IEEE Access,* vol. 10, pp. 14120-14133, 2022.

[14] W. R. W. M. Razin, T. S. Gunawan, M. Kartiwi, and N. M. Yusoff, "Malaria Parasite Detection and Classification using CNN and YOLOv5 Architectures," in *2022 IEEE 8th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, 2022: IEEE, pp. 277-281.

[15] S. Chowdhury, S. Chowdhury, J. T. Ifty, and R. Khan, "Vehicle detection and classification using deep neural networks," in *2022 International Conference on Electrical and Information Technology (IEIT)*, 2022: IEEE, pp. 95-100.