Evaluating Faster R-CNN and YOLOv8 for Traffic Object Detection and Class-Based Counting

Muhammad Talha Jahangir[1], Tahreem Fatima[1], Qandeel Fatima[1]
[1]Institute of Computing, MNS University of Engineering and Technology Multan, Pakistan.
***Correspondence:** mtalhajahangir@mnsuet.edu.pk.

Citation| Jahangir. M. T, Fatima. T, Fatima. Q, "Evaluating Faster R-CNN and YOLOv8 for Traffic Object Detection and Class-Based Counting", IJIST, Vol. 6 Issue. 4 pp 1606-1620, Oct 2024

Received| Aug 22, 2024, **Revised|** Sep 29, 2024, **Accepted|** Oct 03, 2024, **Published|** Oct 10, 2024.

Real-time traffic object detection is a critical component necessary for achieving a fully autonomous traffic system. Traffic object detection, along with background classification, is a significant area of research aimed at enhancing safety on the roads and reducing accidents by accurately identifying vehicles. This research aims to develop an accurate and efficient system for traffic object detection and classification in real-time traffic environments. It also seeks to minimize false positives and negatives, ensuring that no objects are overlooked in the detection of classes such as cars, buses, bicycles, motorcycles, and pedestrians. This research aims and focuses on the two following deep learning technologies: YOLO stands for (You Only Look Once) and Faster R- CNN stands for (Region-based Convolutional neural network). YOLO, initially designed as the single-stage approach, emphasizes speed; therefore, it is best suited for real-time uses. However, Faster R-CNN which is a two-stage detector gives better results in object detection and is highly accurate. Both models are trained and tested on the same data set containing 5712 trained images, 570 validation images, and 270 test images using a workstation with RAM 32 GB and NVIDIA GeForce RTX 4080 Super GPU through the help of CUDA version 12.4 to provide the end evaluating results. Since Faster RCNN is a very intensive model it took 22 hours to complete 3 epochs with an accuracy of 55.2% to train the model and YOLO finished the training within 10 epochs with the mAP@0.5 value of 0.931 of all classes. Our results of traffic object real-time detection indicated that YOLO was vastly better and quicker than Faster R-CNN.

Keywords: Deep learning-based traffic object detection; Autonomous vehicle; Real-time Traffic vision system; Precision and Reliability

Introduction:

The implementation of intelligent traffic management systems and self-driving vehicles act as central components of traffic planning demands precise and high-quality recognition of traffic objects. Deep learning is indeed becoming the game-changing technology in computer vision and its specific application for object detection. Deep learning object detection plays a paramount role coupled with the development of self-driving cars and smart traffic control. These systems depend on the chance of effectively and quickly recognizing itemson the road in real-time. These systems are vital in increasing road safety, observing real-time traffic hence directing traffic, thus decreasing incidences of traffic congestion. Accordingto the facts obtained from the research [1], Road Traffic Accidents (RTA) are recognized to be among the leading causes of injury or death in the world [2]. In addition, the World Health Organization (WHO) [1] provides clear evidence that in 2018, traffic accidents ranked as the eighth leading cause of disease globally, accounting for 2.5% of all deaths worldwide. Similarly,according to WHO statistics, it is also revealed that a ratio of deaths happensevery year. Yu et al. [2] offer an insight into traffic safety concerns, which is a subfield of research in traffic analysis, traffic accident investigation, vehicle–collision identification, collision risk notification, and collision elimination. Additionally, various intelligent systems have been developed to tackle traffic sign detection and recognition using Computer Vision (CV) and Deep Learning (DL) techniques. In thisregard, an effective method for object recognition known as the YOLO (You Only Look Once) object detection algorithm is employed [3] [4] [5] [6] [7] [8] [9] [10] [11].

YOLO represents a more structured system that delivers both high speed and the necessary accuracy for real-time applications. In this way, traffic detection systems can detect traffic patterns that are regularly used and thus precede in recognizing places that presentmore probability of accidents. It enables timely measures like modifying the period for whichthe traffic light remains green or red, changing the existing traffic pattern, or the possibility of using emergency services. For instance, if one intersection records many accidents regularly, traffic detection systems can inform the authorities to act and change either the signs or the signals on the roads for the better. Then, traffic detection systems enhance efficient responses to emergencies as they immediately inform services about accidents. Thus, early identificationand intervention are critical to minimizing the extent of injuries and loss of life. These systems can give the exact location determinants that can help the emergency services to arrive at thescene much faster [3] [4] [5] [6] [7] [8] [9] [10] [11]. Traffic detection systems facilitate data-driven decision-making by supplying the collected data and the urban planners and policymakersto study traffic flows and determine certain behaviors that cause mishaps. Data may reveal that certain types of roads or specific times of day are associated with a higher risk of accidents. This enables authorities to make proper changes in infrastructure that requires it, better road signs including, improved lighting or cycling and walking paths.

Since human error is a significant contributor to many accidents, traffic detection system provides a major relief to drivers if a potential danger arises, these systems provide real-time information to alert drivers and enhance safety. Such systems can alert drivers about slowing down traffic lights, objects on the road, or adverse weather that can lead to accidents. In the most complex scenarios, traffic detection can be integrated with interactions between intelligent and semi-intelligent vehicles, enhancing overall transportation safety by minimizing human error. Summing up, it is possible to state that Traffic detection systems are beneficial for reducing accident rates as they affect traffic openness, security measures, and autonomous automation. This can be financially quantified as a reduction in the toll of lives lost and injuries sustained on the roads, as well as the costs associated with automation.

Our proposed methodology delves into the application of two popular deep learning object detection frameworks: Two of such systems namely YOLO [12] – 'You only look once' and the Faster Region-based Convolutional Neural Networks (CNNs) [13]. These models aim

to identify the classes on a busy traffic road and tally the classes in real-time. Conversely, YOLO, which is built on convolutional neural networks, is one of the models used for object detection. These algorithms in particular have been selected because of their effectiveness and accuracy in contrast with other methods based on DL about the processing time on GPUs, highest results regarding the most crucial values, and alacrity [14], [15], [16]. YOLO is a single-stage detector, making it faster, while Faster R-CNN is a two-stage detector, resulting in greater accuracy. Our objective is to assess these frameworks for detecting various traffic objects using the publicly available Kaggle dataset [17].

This project aims to evaluate the frameworks based on the identification ratio, performance metrics, and their behavior in various environmental conditions. By comparing the strengths and weaknesses of both frameworks and considering the number of classes involved, we seek to establish a foundation for developing an effective traffic object detection system.

Object Detection Models:

Faster RCNN:

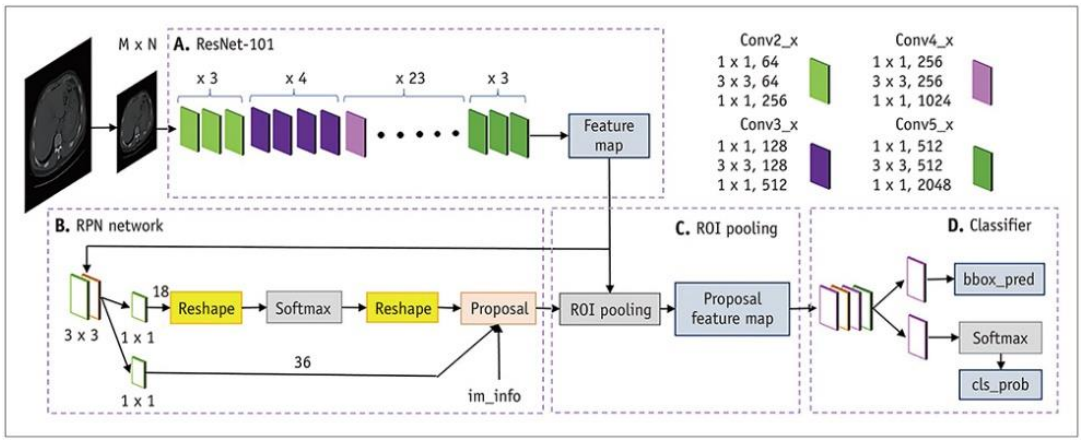

Figure 1. Architecture of Faster R-CNN [18]

Faster R-CNN as in Figure 1 is a deep learning object detection model and one of the latest additions to the object detection family that features a Region Proposal Network (RPN) along with a fully convoluted network. The RPN effectively predicts ranges of feasible object locations in an image; computation time is faster than other techniques, for example, Selective Search. Thus, the fully convolutional network then annotates and regresses these proposals to recognize and localize objects in the image.

YOLO

YOLO (You Only Look Once) is a real-time object detection algorithm. YOLO approach is unique as it processes the image in segments by predicting the bounding boxes and class probabilities for each segment in one pass while other approaches process the images one at a time. It also enables much faster processing rates than other methods of object detection making it suitable for real-time scenarios for example in real-time video analysis and surveillance as well as in AV applications. Another benefit of YOLO's architecture is its capacity to detect multiple objects at once, and therefore offer a holistic perspective of the visual content stream of images and videos.

This paper is divided into three sections, the first section includes a literature review, the second section consists of dataset materials and methods, and the third section includes experimentation and results.

Literature Review

Population growth and increased accessibility to automobiles are contributing to traffic congestion and accidents. Traditional traffic monitoring methods are often labor-intensive, prone to errors, and inadequate for managing large volumes of data. Automated systems that

utilize artificial intelligence and deep learning offer a better solution, delivering rapid and accurate data processing capabilities. However, the creation of a practical system capable of reliably recognizing multiple objects in real traffic environments, while minimizing false positives and false negatives remains a challenging task at this stage. The system's capability to respond efficiently to accidental conditions is essential, as is its ability to adhere to regulatory standards during emergencies. MAA Al-Qaness et al. [19] conducted a study that explores real-time vehicle classification and counting using the YOLOv3 algorithm, utilizing Python libraries such as NumPy, Pandas, and OpenCV. A neural network is trained on over 1,000 [20] images from various datasets (Pascal VOC [21], Open Images [22], COCO [23]) to detect and classify cars, trucks, and buses. The system achieved high vehicle counting accuracy (up to 98%) but experienced seasonal and time-based variations in detection and classification accuracy (e.g., afternoon: 88% detection, 73% classification).

Q Li, X Ding et al [24] utilized YOLOv4 (Python) to detect and count traffic objects in real-time (people, motorcycles, bicycles, cars, buses) to improve road safety for youngsters. The study achieved high accuracy rates for cars (88.89%) and buses (100%), although the detection rates for motorcycles and bicycles were not reported. This limits the model's effectiveness for these important vehicle classes. Additionally, the focus on car and bus accuracy suggested a potential class imbalance in the training data, which might hinder its ability to generalize to other object categories. J. Wang et al. [25] performed a study on traffic sign identification and classification employing YOLOv5, highlighting efficiency and utilizing a constrained dataset. To enhance accuracy, the standard YOLOv5 was upgraded with an improved Adaptive Attention Feature Pyramid Network (AF-FPN). The standard GIoU loss metric was substituted with the more straightforward Intersection over Union (IoU) for assessment, whereas the primary performance parameter was Frames Per Second (FPS). Achieving high FPS was tough due to the restricted training dataset, comprising merely 2,000 images. Dependence on established data augmentation techniques may insufficiently encompass the complete spectrum of changes present in actual traffic signs.

P Shinde et al [26] proposed a solution using real-time traffic light control. Cameras and YOLO were used to count vehicles and adjust lighting based totally on lane density and wait instances. This aimed to reduce congestion and emissions. However, the accuracy of YOLO for vehicle detection is probably limited, specifically for smaller gadgets or in difficult conditions. The accuracy of YOLO for vehicle detection in this scenario is unknown, and it might prioritize speed over precision, especially for smaller objects or lower-quality footage. Despite these limitations, YOLO offers a promising step towards smarter traffic light control. N Youssouf - Heliyon et al [27] proposed a Convolutional Neural community (CNN) for traffic sign recognition and a YOLOv4-based object detection system for real-time traffic sign detection. The Faster R-CNN achieved an accuracy of 43.26% on the GTSRB dataset. The YOLOv4 model achieved a mean average precision (map) of 59.88% at a frame rate of 35 fps. The CNN for traffic sign recognition, although achieving high accuracy, requires

During these kinds of events, processing time is very important e.g., it takes almost 6.63 seconds to classify all images in the test set, making it slow for real-time applications. Faster R-CNN is slow (6 fps) for real-time applications. V Dalborgo et al [28] explored using YOLOv5, a deep learning model, to detect and classify traffic signs in Brazil, with a particular focus on signs obscured by vegetation. They compared different YOLOv5 versions (n, m, and x) and found that YOLOv5m offered the best compromise. YOLOv5m achieved good overall detection accuracy while maintaining a similar performance to the more complex YOLOv5x. Importantly, it excelled at detecting vegetation occlusion with a precision of 92.9%. The researchers trained and tested their model using a dataset of 5631 images with 8065 annotations, encompassing 16 different traffic sign classes along with a specific class for vegetation occlusion. The hardware used for evaluation (i5 CPU, 16GB RAM, GTX920M

GPU) might not be ideal for real-time applications. More powerful hardware is recommended.

Table 1. Literature review and study.

Sr. no	Research Paper	Method	Classes	Results
1	An improved YOLO-based road traffic monitoring system	YOLOv3 detects vehicles. Python, NumPy, pandas, & OpenCV. An image dataset of ten thousand surveillance cameras is used in training a neural network.	Detect cars, trucks, and buses.	Afternoon: detection, 88 %; classification, 73 %; counting, 98%; IoU, 98%. Night: detection accuracy of 82%, classification accuracy of 65%, counting accuracy of 94% and IoU of 86%. Winter: detection: 67%; classification: 49%; counting: 85%; IoU: 81%.
2	Detection and identification of moving objects on busy traffic roads based on YOLOv4	YOLOv4 is used. Python is used. VOC and UA-DETAC datasets are used.	Person, Motorcycle, bicycle, car and bus	Detection accuracy: person (75%), bike (0), bicycle (0), car (88.89%), bus (100%).
3	Improved YOLOv5 network for real-time multi-scale traffic sign detection	YOLOv5 with improved AF-FPN is used to detect vehicles. FPS is the evaluation metric. 2000 images are used. The loss function used is changed from GIoU to IoU.	Detect only different traffic signs.	Good changes in traffic management i.e. reliable but slow with the mAP of 0.6514
4	Smart traffic control system using YOLO	YOLO detects vehicles. Raspberry Pi together with AWS for deployment. Wi-Fi for video transfer. Background detector for lane density measurement applied to Intelligent Traffic Control. Internet connection required.	Detect cars, trucks, and bikes.	Accuracy is not given. Used for counting
5	Traffic sign classification using CNN and detection using faster-RCNN and YOLOV4	YOLOv4 for traffic sign recognition. GTSRB dataset. 0. 8 million parameters. Fine-tuned on GTSRB for 43 classes using weights of Faster R-CNN. Testing data is 20% and validation data 20%, the training data takes 60%.	traffic signs (also German)	accuracy is 99.20%
6	Traffic sign recognition with deep learning (vegetation occlusion	YOLOv5 deployed for traffic sign detection. YOLOv5m is best. Here we have 5631 images with 8065 non-annotated images used during the	Detect only different traffic signs.	Accuracy of Faster R-CNN .43.26%. Map of YOLOv4 is 59.88%

	detection in the Brazilian environment)	study. There are 16 classes of traffic signs + vegetation occlusion is detected.		
7	Ours	Faster R-CNN and YOLO models are chosen as evaluation models. The Kaggle dataset is used with 5712 trained images.	Car, bus, bicycle, person and motorbike	Faster R-CNN gave 55.3% accuracy in 22.1 hours in 3 epochs and YOLO gave 93.1% mAP in 10 epochs.

Methodology:

This section outlines the process of integrating data into the traffic object detection project using YOLOv8 or Faster R-CNN. The data is sourced from Kaggle [17], an online platform that provides access to a wide variety of datasets for machine learning projects. Since our methodology emphasizes transportation items, meticulous attention is required for data cleansing and preparation. This entails verifying that the many class labels in the annotations correspond with both general and specific categories, including vehicles, buses, bicycles, motorbikes, and pedestrians. By redefining and reorganizing some of this data, the study aims to provide an enhanced dataset that better suits the models designed for analyzing and detecting these traffic objects. To effectively train and evaluate the object detection model, the dataset is typically divided into three distinct subsets: As with the previous methods, this has three stages namely training, validation, and testing. The training set is employed to bring the model into contact with many labeled examples to establish good features concerning different object classes. The validation set proves useful during the learning process as it offers information on the model's performance and assists in tuning a model that has suffered from overfitting, a situation whereby a model gets overly trained to the extent that it cannot generalize to other examples. Lastly, the testing data allows for the evaluation of the model's accuracy on unseen data and can be considered as a rather realistic evaluation of its performance. Our model is designed to detect similar scenarios and situations much like the ones represented within the provided dataset, limiting its applicability to such contexts. The dataset used is described in Figure 2.

Figure 2. The dataset used in the model.

The dataset includes 5,712 train images, 541 validation images, and 270 test images. The model is trained on a workstation with RAM 32 GB and NVIDIA GeForce RTX 4080 Super GPU. To enhance computational efficiency, the device used CUDA model 12.4. This setup enabled effective utilization of the GPU, appreciably accelerating the trained and evaluation procedures. The combination of high RAM and a powerful GPU provided robust performance throughout the development of the version. We also purchase Colab [29] two times to run heavy models like Faster RCNN and YOLO.

The initial step involves converting the dataset labels to the Pascal VOC format for Faster R-CNN (Figure 3) because it standardizes annotations, ensuring compatibility with pre-trained models. Pascal VOC organizes dataset labels into XML files containing item categories and bounding box coordinates, which manual the version in detecting and classifying objects throughout training and evaluation. Data preprocessing is a vital segment, concerning resizing

pix to smaller dimensions, standardizing pixel depth values, and making use of numerous augmentation strategies inclusive of cropping, flipping, or color jittering to beautify the robustness of the version. A pre-skilled convolutional neural backbone ResNet50, asserted a vital role in deriving huge capabilities from a given image input. As a model, ResNet50, pretrained at the COCO dataset familiar with taking input characteristics including edges, shape, and texture that are vital for other detection levels. In this particular case, the Region Proposal Network (RPN) is held responsible for generating capability regions of interest that would contain objects. It works by moving a window over the feature maps that are generated using the spine network to locate the possible areas that contain devices.

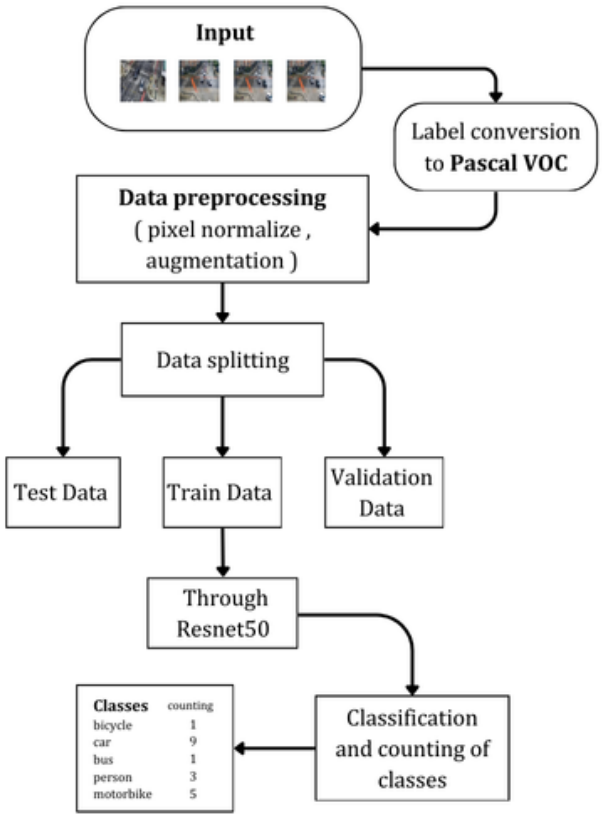

Figure 3. Faster RCNN architecture methodology of traffic object detection

Following this, ROI pooling was employed to manner the proposed areas from the RPN, changing them into constant-size feature maps. This step is important as it standardizes the scale of inputs for the subsequent layers, ensuring consistency and efficiency in processing. Finally, classification and bounding container regression are completed. The community performs these tasks simultaneously: classifying each region as either an object or background and refining the boundaries of those areas to enhance the accuracy of the bounding boxes. Classification is normally treated using a SoftMax layer, which assigns probabilities to one-of-a-kind item lesson while bounding box regression is managed using a regression layer, which adjusts the coordinates of the bounding containers to higher healthy the detected gadgets. This combined technique enhances both the detection accuracy and the precision of the object localization in the image.

$$tx = (x - xa)/wa \ , \ ty = (y - ya)/ha$$
$$tw = \log(w/wa) \ , \ th = \log(h/ha)$$

Where x, y, w, h represents the predicted box's center coordinates, width, and height, and xa, ya, wa, ha represents the anchor box's center coordinates, width, and height. The classification loss for object detection in Faster R-CNN is often based on cross-entropy loss.

$$Lcls(pi, pi*) = -\sum_{i=1}^{N} pi * \log(pi)$$

Where pi is the predicted probability of the response being equal to the desired value given the values of all the predictors. Iwith class, and Pi∗ represents the ground truth label; if the current instance belongs to the correct class, then the ground truth label is 1, else it is 0. The bounding box regression loss is normally derived from the smooth L loss:

$$L_{reg}(ti, ti*) = smooth_{L_1}(ti, ti*)$$

The results are provided in the form of detected bounding boxes for each class, class label, and class-based counting.

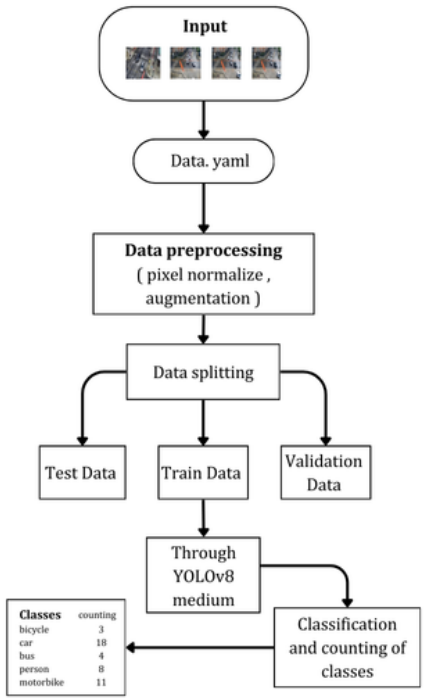

Figure 4. YOLO architecture methodology of traffic object detection

YOLO (You Only Look Once) is best for real-time packages due to its capability to speedy process snapshots while shooting contextual data [3] [4] [5] [6] [7] [8] [9] [10] [11]. To configure YOLO for effective training and validation, we make use of a configuration report, normally with Yaml extension, which details paths to the training and validation datasets, specifies the image size, and units' other parameters necessary for version operation. Data preprocessing is an important step in which the input images go through several transformations to optimize the model's performance. This includes resizing pictures to a fixed size for consistency, standardizing pixel intensities to normalize the data, and making use of augmentation strategies including cropping, flipping, and color balancing. These preprocessing steps are designed to beautify the robustness and generalization of the version, ensuring accurate item detection across numerous situations. The dataset is then split into 3 distinct subsets: training, validation, and testing. This division is vital for comparing the model's overall performance and making sure that it generalizes properly to new, unseen information. During the training segment, ResNet-50, a neural community pre-educated on thousands and thousands of photos, is hired. ResNet-50 [30] is adept at spotting styles and capabilities within images, making it effective backbone for training YOLO in our dataset. Finally, YOLO performs bounding box prediction and class probability estimation. The model simultaneously predicts bounding containers and the associated class probabilities for each grid mobile within

the image or input. This dual prediction capability allows YOLO to accurately determine the places of objects and classify them inside the image. For every bounding box, YOLO presents predictions concerning its position and the probability of each class, enabling precise object detection and classification. Center Coordinates (x, y):

$$x = \sigma(x^A) + cx \quad , \quad y = \sigma(y^A) + cy$$

Where x^and y^ are the predicted coordinates, cx, and cy are the coordinates of the cell which is a part of the grid. σ is commonly denoted sigmoid function or logistic function. Width (w) and Height (h)

$$w = pw.\exp(w^A) \quad , \quad h = ph.\exp(h^A)$$

Where w^ and h^ are the expected width and peak, and pw and ph are anchor box dimensions. YOLO predicts confidence based on the IoU score for each bounding box.

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union}$$

Where: Intersection over Union is the ratio of the Area of Overlap to the union of the predicted bounding box and the truth bounding box. Mathematically, it can be expressed as:

$$IoU = |\ A \cap B\ |/|\ A \cap B\ |$$

Where A is the predicted bounding box. B is the truth bounding box. |A∩B | represents the area of overlap between the predicted and truth-bounding boxes [31]. |AUB| represents the area of union which can be calculated as the sum of the areas of the predicted and truth boxes minus the area of overlap.

$$IoU = \frac{(Tp)}{(Tp) + (Fn) + (Fp)}$$

Where Tp is true positive, Fn is false negative and Fp is false positive. Non-maximum suppression helps minimize the false positive results and at the same time increases the overall likelihood of detection. The next phase which is the training loop entails going through the entire process from the input image to the weights' update several times in the training data set. Such considerations enable the model to undergo iterations that enhance the object detection it possesses. After training the model, a certain data set is used to test the effectiveness and reliability of the model. It is crucial to measure the accuracy; thus, several evaluation metrics like Mean Average Precision (mAP) are used. Mean Average Precision is another important measure used to calculate the performance of an object detection model since it measures both precision which is the number of objects correctly detected, and recall which is the proportion of all objects that have been detected with Intersection over Union threshold. For every class, obligatory metrics Average Precision (AP) enables the Precision-Recall curve integration.

$$Precision = (Tp)/(Tp) + (Fp)$$

$$AP = \int_0^1 P(r)dr$$

The mAP is calculated by finding Average Precision (AP) for each class and then averaging over a few classes for results.[33]

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi$$

Experimentation and results

In the initial run, 55% of the images were correctly recognized, indicating the need for

similar training through additional epochs to enhance the model's overall performance. Training the Faster R-CNN model over three epochs took about 22.15 hours, highlighting each computational demand for and time for better optimization.

The evaluation of the Faster R-CNN model revealed blended performance across distinctive item instructions and achieved an affordable accuracy of 55.24%, it played well in general in detecting vehicles, with a precision of 0.6457 and an F1 Score of 0. 6687. The model struggled to recognize bikes and buses, which were not accurate, memory and F1 points. Also not recognize motorcycles and persons. This means that although the model is effective for some applications, such as cars, it needs more refinement and more balanced training data to improve its overall performance.

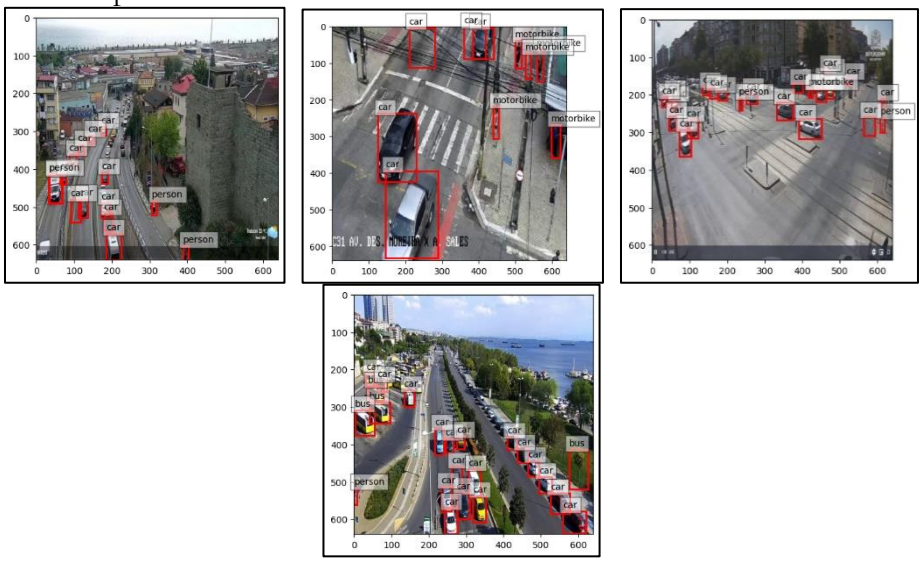

Figure 5. Detected results of Faster RCNN

Table 2. Precision, Recall, and F1 score of Faster RCNN

Metric	Precision	Recall	F1 score
Bicycle	0	0	0
Bus	0.0435	0.0488	0.046
Car	0.6457	0.6934	0.6687
Motorbike	0.3818	0.4123	0.3964
Person	0.4278	0.3673	0.3952

After implementing the YOLOv8 medium model, it was tested on the testing images. The model was trained for 10 epochs in 0.208 hours. During the initial testing, 93% of the images were recognizable, indicating that further training over additional epochs could enhance the program's results. The evaluation of the model shows strong overall performance universal, with a mAP50 (suggest Average Precision at 50% IoU) of 0.931 and a mAP50-95, indicating the model's effectiveness in detecting and appropriately localizing classes.

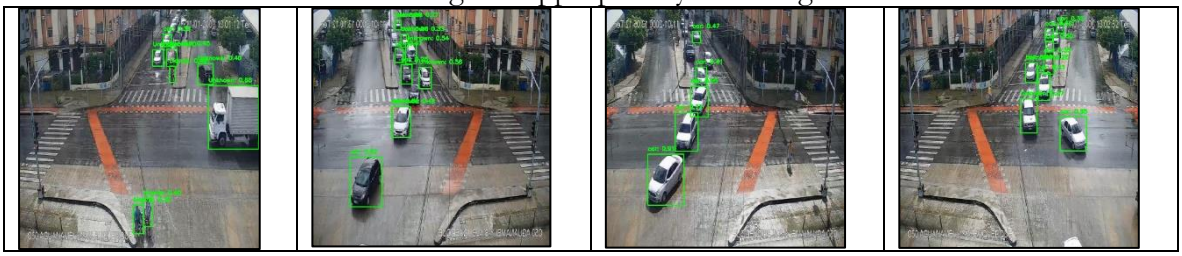

Figure 6. Detected results of YOLO

The model performed particularly nicely with cars, buses, and bicycles, and achieved high precision reflected in their mAP50 rankings of 0.965 and 0.956, respectively. For bicycles

and motorbikes, the model indicated solid overall performance, with mAP50 ratings of 0.950 and 0.904, even though the precision and recollect were slightly lower, for persons indicating a little room for improvement in detecting those training. Overall, at the same time as the version demonstrated high performance with cars, mainly bicycles, and buses, it showed regions for improvement in detecting and localizing motorbikes and persons.

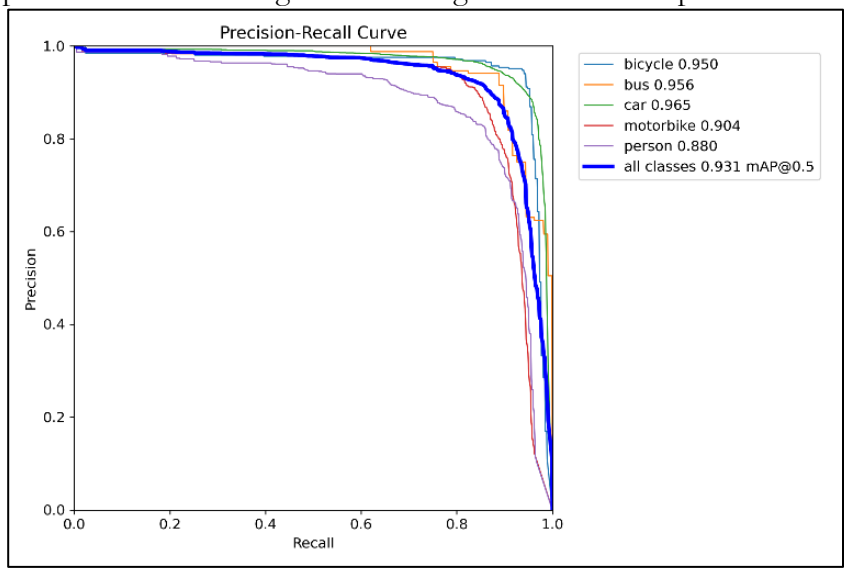

Figure 7. Precision- Recall curve of YOLO

The results indicate that the YOLO performs well and accurately, as it correctly sthe class in the diagonal elements. Off diagonal represents the misclassification of classes like 9 cars were classified incorrectly as buses. The graphs in Figure 9. show that the decreasing training and validation loss curves indicate that the model is getting better at learning the data over time. Precision and recall values improved over time, indicating that the model is able to recognize objects with a low number of false alarms. Additionally, the mAP50 and mAP50-95 metrics demonstrate that the model performs well overall in object detection. The increase in these metrics indicates that the model is improving its ability to detect and classify objects as we transition from one IoU threshold to another.

In comparison between YOLOv8 and Faster R-CNN, YOLO's single-stage architecture offers superior pace and accuracy, converging at some point of training, with improving performance metrics. However, there is still potential for improvement, particularly regarding the mAP50-95 metric. Potential areas for development include growing schooling epochs, adjusting hyperparameters, information augmentation, and exploring exclusive model architectures. By addressing these areas, the model's performance can be similarly more advantageous.

Faster R-CNN, despite its mild accuracy, is computationally pricey, limiting its suitability for real-time packages due to making it a greater sensible choice for real-time visitor's item detection. However, to draw a definitive conclusion, it is essential to evaluate the overall performance specific to each class.

While each way exhibits comparable overall accuracy, YOLO and Faster R-CNN performance may additionally vary substantially while detecting particular item classes. A complete expertise of their strengths and weaknesses can be gained by cautiously analyzing their overall performance across special object lessons, enabling a knowledgeable selection for the maximum appropriate version in visitor's item detection packages.

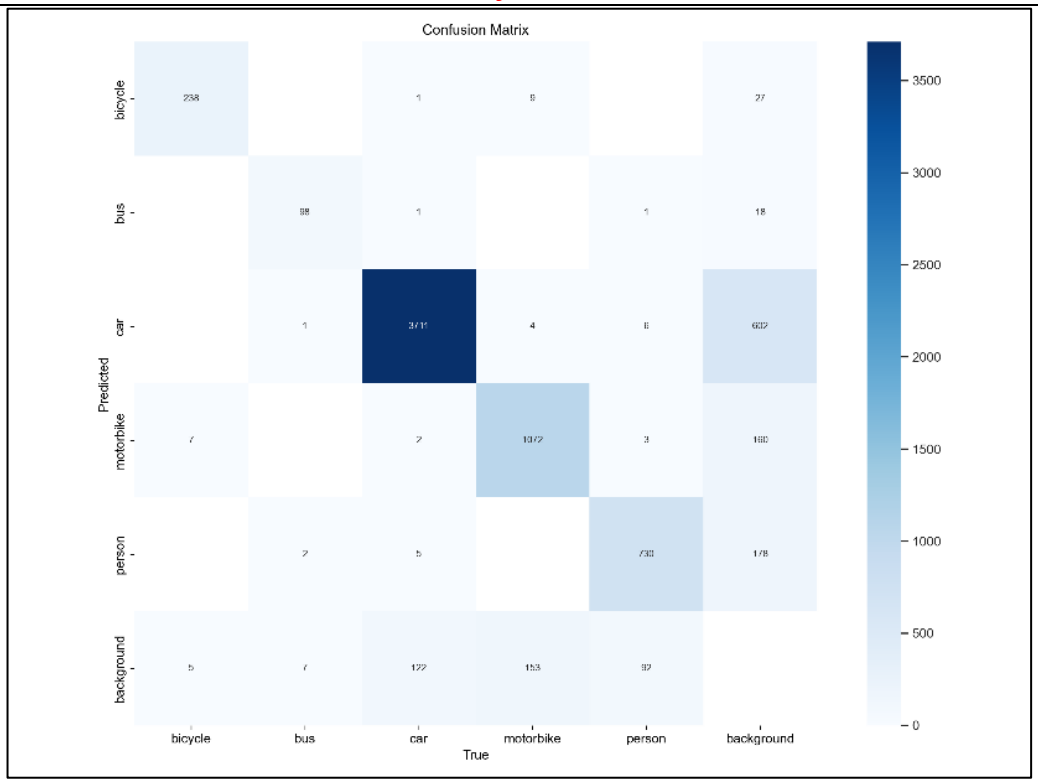

Figure 8. Confusion matrix of YOLO

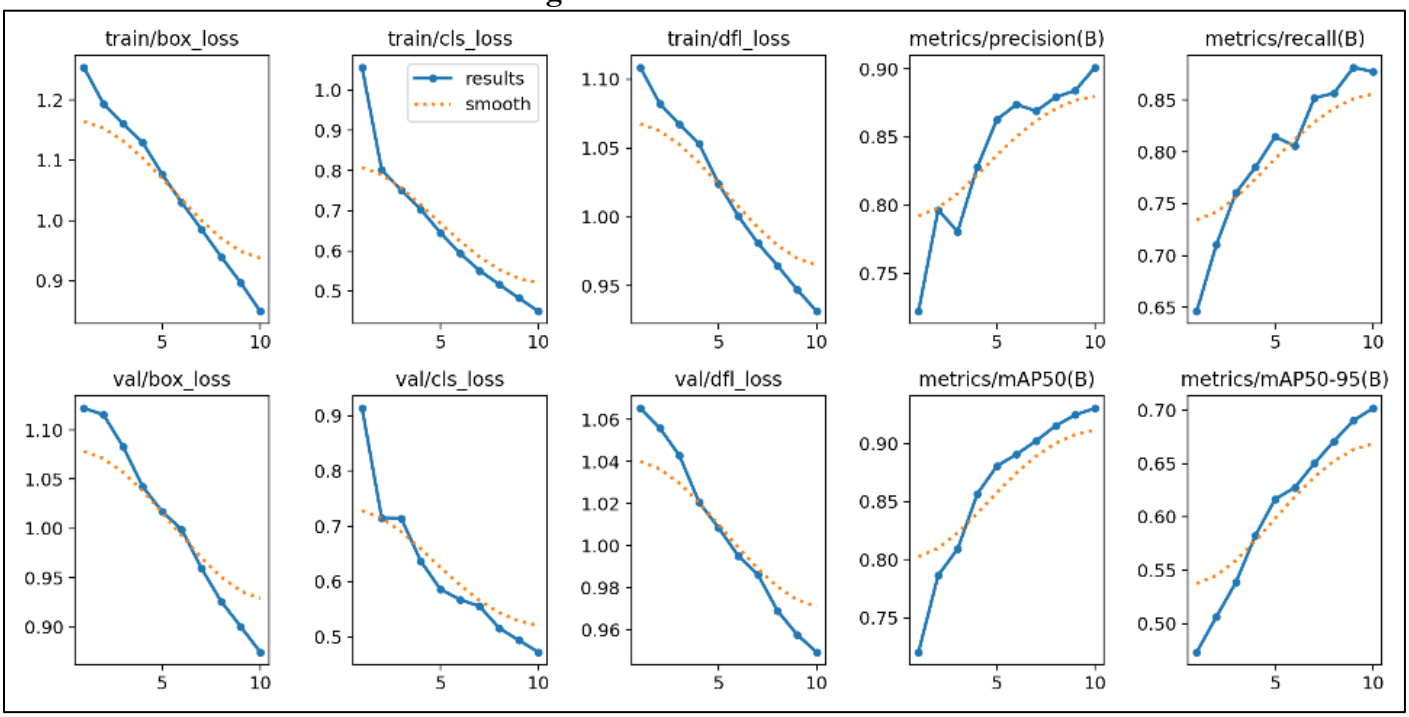

Figure 9. Graph of detected output in YOLO

Discussion.

As our methodology explains the evaluation between Faster R-CNN with an accuracy of 55% completed 3 epochs in 22 hours as a very extensive model and YOLOv8 with an accuracy of 93% in 0.208 hours. We choose YOLOv8, as YOLOv8 is better than YOLOv5, and YOLOv7 in terms of architecture, feature extraction, computational time, and performance of objects smaller. It also provides more capabilities in post-processing, options for customizations, and compatibility with the recent frameworks as compared to the generic filters

making it more effective.

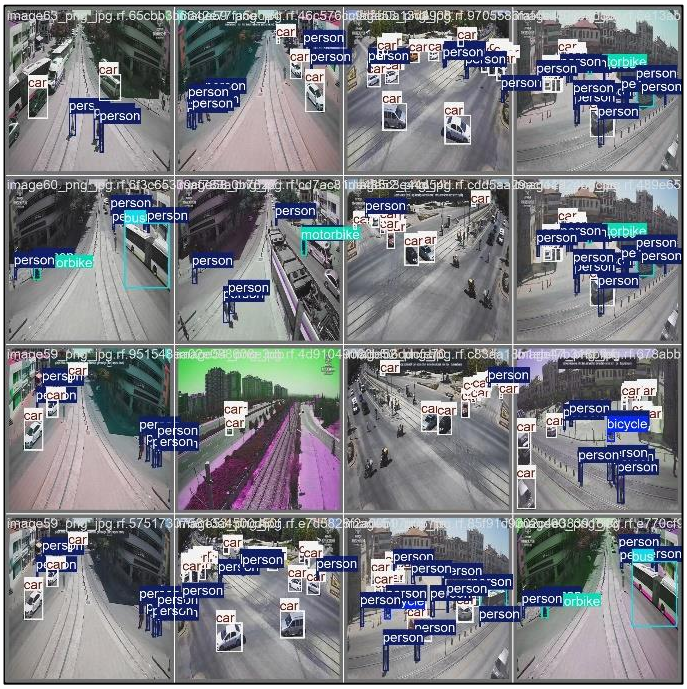

Figure 10. Detected output in YOLO with labels

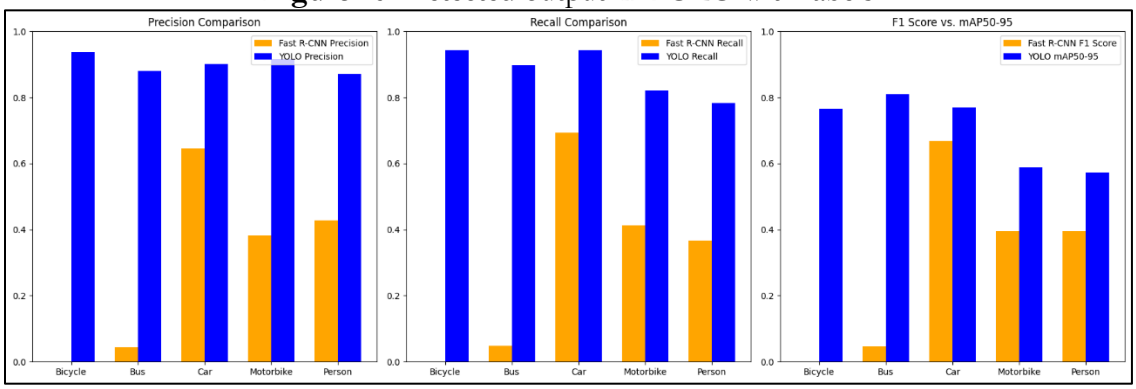

Figure 11. Comparison of Faster RCNN and YOLOv8 medium Results of Faster RCNN.

Conclusion and Future Work:

In conclusion, this study assessed the overall performance of Faster R-CNN and YOLOv8 for real-time traffic detection, emphasizing the change-off between pace and accuracy. Faster R-CNN accomplished 55% accuracy after three epochs, presenting robust detection however restrained real-time applicability because of its computational demands. In evaluation, YOLOv8 reached 93% accuracy after 10 epochs in 0.206 hours, making it more suitable for actual-time use, even though with potentially lower accuracy. Both models were trained on a Kaggle dataset containing 5,712 images, utilizing a workstation equipped with 32 GB of RAM and an NVIDIA GeForce RTX 4080 Super GPU, powered by CUDA version 12.4. Additionally, we can enhance the results by exploring other deep-learning technologies. In the future, we will work for the better detection of bicycles and pedestrians by using Efficient and SSD

Acknowledgment. The authors thank MNS UET for letting us use their NVIDIA GeForce RTX 4080 Super GPU.

Author's Contribution. This research is a team work and all authors contributed equally.

Conflict of interest. The authors declare no conflict of interest regarding the publication of the paper.

Project details. Nil

References:

[1] Organização Mundial da Saúde, "GLOBAL STATUS REPORT ON ROAD SAFETY 2018 SUMMARY," World Heal. Organ., no. 1, p. 20, 2018, Accessed: Sep. 30, 2024. [Online]. Available: http://apps.who.int/bookorders.

[2] G. Yu, P. K. Wong, J. Zhao, X. Mei, C. Lin, and Z. Xie, "Design of an Acceleration Redistribution Cooperative Strategy for Collision Avoidance System Based on Dynamic Weighted Multi-Objective Model Predictive Controller," IEEE Trans. Intell. Transp. Syst., vol. 23, no. 6, pp. 5006–5018, Jun. 2022, doi: 10.1109/TITS.2020.3045758.

[3] C. Liu, S. Li, F. Chang, and Y. Wang, "Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives," IEEE Access, vol. 7, pp. 86578–86596, 2019, doi: 10.1109/ACCESS.2019.2924947.

[4] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo Algorithm Developments," Procedia Comput. Sci., vol. 199, pp. 1066–1073, Jan. 2022, doi: 10.1016/J.PROCS.2022.01.135.

[5] S. P. Rajendran, L. Shine, R. Pradeep, and S. Vijayaraghavan, "Fast and Accurate Traffic Sign Recognition for Self Driving Cars using RetinaNet based Detector," Proc. 4th Int. Conf. Commun. Electron. Syst. ICCES 2019, pp. 784–790, Jul. 2019, doi: 10.1109/ICCES45898.2019.9002557.

[6] W. Yang and W. Zhang, "Real-Time Traffic Signs Detection Based on YOLO Network Model," Proc. - 2020 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2020, pp. 354–357, Oct. 2020, doi: 10.1109/CYBERC49757.2020.00066.

[7] C. Dewi, R. C. Chen, Y. T. Liu, X. Jiang, and K. D. Hartomo, "Yolo V4 for Advanced Traffic Sign Recognition with Synthetic Training Data Generated by Various GAN," IEEE Access, vol. 9, pp. 97228–97242, 2021, doi: 10.1109/ACCESS.2021.3094201.

[8] D. Mijic, M. Brisinello, M. Vranjes, and R. Grbic, "Traffic Sign Detection Using YOLOv3," IEEE Int. Conf. Consum. Electron. - Berlin, ICCE-Berlin, vol. 2020-November, Nov. 2020, doi: 10.1109/ICCE-BERLIN50680.2020.9352180.

[9] W. N. Mohd-Isa, M. S. Abdullah, M. Sarzil, J. Abdullah, A. Ali, and N. Hashim, "Detection of Malaysian Traffic Signs via Modified YOLOv3 Algorithm," 2020 Int. Conf. Data Anal. Bus. Ind. W. Towar. a Sustain. Econ. ICDABI 2020, Oct. 2020, doi: 10.1109/ICDABI51230.2020.9325690.

[10] A. Avramović, D. Sluga, D. Tabernik, D. Skočaj, V. Stojnić, and N. Ilc, "Neural-network-based traffic sign detection and recognition in high-definition images using region focusing and parallelization," IEEE Access, vol. 8, pp. 189855–189868, 2020, doi: 10.1109/ACCESS.2020.3031191.

[11] Y. Gu and B. Si, "A Novel Lightweight Real-Time Traffic Sign Detection Integration Framework Based on YOLOv4," Entropy 2022, Vol. 24, Page 487, vol. 24, no. 4, p. 487, Mar. 2022, doi: 10.3390/E24040487.

[12] "YOLO Algorithm for Object Detection Explained [+Examples]." Accessed: Sep. 30, 2024. [Online]. Available: https://www.v7labs.com/blog/yolo-object-detection

[13] "Faster R-CNN Explained for Object Detection Tasks | DigitalOcean." Accessed: Sep. 30, 2024. [Online]. Available: https://www.digitalocean.com/community/tutorials/faster-r-cnn-explained-object-detection

[14] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja, "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues," Array, vol. 10, p. 100057, Jul. 2021, doi: 10.1016/J.ARRAY.2021.100057.

[15] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," Multimed. Tools Appl., vol. 82, no. 6, pp. 9243–9275, Mar. 2023, doi: 10.1007/S11042-022-13644-Y/TABLES/7.

[16] A. Boukerche and Z. Hou, "Object Detection Using Deep Learning Methods in Traffic Scenarios," ACM Comput. Surv., vol. 54, no. 2, Mar. 2021, doi: 10.1145/3434398.
[17] "Kaggle: Your Home for Data Science." Accessed: Sep. 30, 2024. [Online]. Available: https://www.kaggle.com/datasets/yusufberksardoan/traffic-detection- project/data
[18] "kjr-21-869-g002-l.jpg (1005×407)." Accessed: Sep. 30, 2024. [Online]. Available: https://kjronline.org/ArticleImage/0068KJR/kjr-21-869-g002-l.jpg
[19] M. A. A. Al-qaness, A. A. Abbasi, H. Fan, R. A. Ibrahim, S. H. Alsamhi, and A. Hawbani, "An improved YOLO-based road traffic monitoring system," Computing, vol. 103, no. 2, pp. 211–230, Feb. 2021, doi: 10.1007/S00607-020-00869-8/METRICS.
[20] "How many images do you need to train a neural network? « Pete Warden's blog." Accessed: Sep. 30, 2024. [Online]. Available: https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/
[21] "The PASCAL Visual Object Classes Homepage." Accessed: Sep. 30, 2024. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/
[22] "Open Images V7." Accessed: Sep. 30, 2024. [Online]. Available: https://storage.googleapis.com/openimages/web/index.html
[23] "COCO - Common Objects in Context." Accessed: Sep. 30, 2024. [Online]. Available: https://cocodataset.org/#home
[24] R. Chavan, A. Gulge, and S. Bhandare, "Moving Object Detection and Classification using Deep Learning Techniques," Tuijin Jishu/Journal Propuls. Technol., vol. 45, no. 01, pp. 3935–3944, Feb. 2024, Accessed: Sep. 30, 2024. [Online]. Available: https://www.propulsiontechjournal.com/index.php/journal/article/view/5000
[25] J. Wang, Y. Chen, Z. Dong, and M. Gao, "Improved YOLOv5 network for real-time multi-scale traffic sign detection," Neural Comput. Appl., vol. 35, no. 10, pp. 7853–7865, Apr. 2023, doi: 10.1007/S00521-022-08077-5/METRICS.
[26] P. Shinde, S. Yadav, S. Rudrake, and P. Kumbhar, "Smart Traffic Control System using YOLO," Int. Res. J. Eng. Technol., 2019, Accessed: Sep. 30, 2024. [Online]. Available: www.irjet.net
[27] N. Youssouf, "Traffic sign classification using CNN and detection using faster-RCNN and YOLOV4," Heliyon, vol. 8, no. 12, Dec. 2022, doi: 10.1016/j.heliyon.2022.e11792.
[28] "Welcome To Colab - Colab." Accessed: Sep. 30, 2024. [Online]. Available: https://colab.research.google.com/
[29] V. Dalborgo et al., "Traffic Sign Recognition with Deep Learning: Vegetation Occlusion Detection in Brazilian Environments," Sensors 2023, Vol. 23, Page 5919, vol. 23, no. 13, p. 5919, Jun. 2023, doi: 10.3390/S23135919.
[30] "Roboflow Blog." Accessed: Sep. 30, 2024. [Online]. Available: https://blog.roboflow.com/what-is-resnet-/
[31] "Intersection over Union (IoU): Definition, Calculation, Code." Accessed: Sep. 30, 2024. [Online]. Available: https://www.v7labs.com/blog/intersection-over-union-guide