Occlusion-aware Module for 2D Object Detection for Autonomous Driving

Alauddeen Shaikh
alauddeensha@umass.edu

Amit Sarker
asarker@umass.edu

Roshini Pulishetty
rpulishetty@umass.edu

Abstract

Accurate object detection is critical for safe autonomous driving, yet challenges like partial occlusions, false positives, and misclassifications hinder existing frameworks' performance. We propose a two-stage object detector that enhances YOLOv5m with ResNet-18 secondary classifier and further, show that overlap with initial predictions (IoU) and aspect ratios serve as important information for predicting occluded parts. The secondary classifier refines YOLOv5m's predictions by reducing false detections, while the occlusion-aware module improves the localization of partially occluded objects. Evaluated on a vehicle dataset derived from MS COCO, our model outperforms YOLOv5 baselines, achieving higher mean Average Precision (mAP) and Average Recall (AR) across all vehicle categories, especially for small and medium-sized objects.

1. Introduction

Accurate object detection is crucial for the safety and reliability of autonomous driving systems and advanced driver-assistance systems (ADAS) [13]. This involves accurately identifying and locating objects in an image or video. One of the primary challenges of object detection is *occlusion*, where an object is partially visible in the frame due to another object blocking its view. Partial occlusions lead to missed detections or inaccurate localizations. This degrades the performance of critical downstream tasks, such as object tracking and path planning, leading to serious consequences of road accidents and risky robot moves. Additionally, the false positive rate of incorrectly identifying an object is too high in the existing detection frameworks. Hence, this study aims to enhance the robustness of object detectors to partial occlusions in autonomous driving scenarios.

Problem Statement. Given an RGB image $I \in \mathcal{R}^{H\times W\times 3}$, object detection aims to regress the bounding box of each object of interest $B = (x_{center}, y_{center}, w, h)$ in the image I and classify them into respective object classes, along with assigning a confidence score to each detection. The main goals of this project are twofold: (1) to produce accurate localizations of objects during occlusions, predicting the complete bounding box even when the object is partially visible in the scene, and (2) to reduce false positives and misclassifications in the detection of vehicles.

We address this problem by developing a two-stage detection pipeline that integrates a secondary classifier into the YOLO (You Only Look Once) [9]. We chose the YOLOv5m model as it is a one-stage detector with a real-time inference system while performing on par with two-stage detectors in accuracy. For this, we design and implement a secondary classification stage using a ResNet-18 architecture, which validates and refines the initial detections made by YOLOv5m.

When a vehicle is partially obscured, the algorithm must compensate for the missing information for accurate detection. This requires a multi-faceted approach that can effectively adjust bounding boxes to account for occlusions and validate detections to reduce false positives. We hypothesize that by integrating a secondary classifier to refine detections and implementing post-processing enhancements for occluded bounding boxes, we can significantly improve the detection performance of vehicles in challenging scenarios.

We started with investigating how YOLOv5m performs in occlusion scenarios after fine-tuning with the vehicle COCO dataset [8], analyzing the gaps where it produces false positives and the impact of occlusions. The main contributions of this work are:

1. We propose a two-stage detection pipeline that integrates a ResNet-18 secondary classifier into the YOLOv5m framework, which refines detection predictions and reduces false positives and misclassifications.
2. We implement an occlusion-aware bounding box enhancement method that scales occluded bounding boxes based on IoU and proximity gaps, improving the localization accuracy of partially occluded objects without retraining the model.
3. We demonstrate that our model achieves better detection accuracy while maintaining computational efficiency across the YOLOv5 family.

2. Related work

Recent advancements in 2D object tracking have significantly enhanced the performance of tracking on vehicles and pedestrians. Transformer-based architectures, such as SwinTrack [7], tremendously improved the performance by leveraging hierarchical feature representations. While there are several frameworks developed to tackle occluded objects, like TransMOT [3] utilizes spatial-temporal graph transformers to effectively model object relationships across frames, FairMOT [14] jointly optimizes object detection and re-identification into a unified network and ByteTrack [15] which recovers the high-score and low-score detections using different association metrics, there is still a scope of improvement.

R-CNN family of models [5] have revolutionized the field of object detection using region proposals and convolutional neural networks. These object detectors employ two stages of detection - Region Proposal Network (RPN) proposes candidate object bounding boxes, and the second stage extracts features from each proposal and determines the bounding box and class it belongs to. As they employ selective search to propose regions and pass each of them through CNNs, which is computationally expensive, it can not be applied in object tracking which requires fast inference and a complex training pipeline. Later, there were variants of R-CNNs, such as fast R-CNNs [4], faster R-CNNs [10], and fastest R-CNNs. Fast R-CNNs [4] builds on the limitations of R-CNNs by optimizing region proposals, by running CNNs and performing RoI only once instead of running it several times. However, even after 20x speed improvement during inference time over R-CNNs, they did not achieve real-time inference due to heavy dependency on region proposal generation.

Another variant of R-CNN, Cascade R-CNNs [2] uses a multi-stage framework to localize, allowing it to refine the bounding box progressively over the stages. This allows the network to accurately detect even in the presence of major occlusions and is frequently used in high-accuracy applications in autonomous driving and surveillance. However, this can not be used in real-time tracking of video sequences, as this network does not infer in real-time and is iterative. Built on R-CNNs, Sparse R-CNNs [12] handle occlusions with a query-based approach, where "object queries" directly interact with feature maps allowing it to focus on localized features even when occluded. It is shown that it works well in dense and cluttered scenes, and is deployed in 3D object tracking of SRCN3D [11]. However, it leverages 3D surround view of camera data for detection, while our requirement is for the module to work with a single image of each scene.

With a similar aim of localization, TridentNet [6] uses multiple parallel branches of different receptive fields so as to capture objects of different sizes. Though it has not been studied extensively, this multi-branch design could potentially assist the model in handling occlusions by merging information from different receptive fields. To make the lighter networks for object localization, CenterNet [16] is a key point-based approach to detect the key points at the center of the bounding box. While this network could localize objects in real-time, it can potentially produce wrong results when the whole side of an object is missing, that is in major occlusions, since it treats each object as a point.

YOLO (You Only Look Once) [9] is the single object detection module and one of the most efficient approaches that applies a single-stage framework for fast detection. Due to significant improvements in feature extraction and object localization, they are integrated with many other frameworks where localization is an auxiliary task. While being rapid for real-time tracking, they overcome occlusions as well to some extent. However, they can not tackle major occlusions or when the object has been cut from the image on one side.

3. Method

With a goal to improve accuracy and reduce false positives in object recognition tasks, we developed a two-stage detection pipeline that integrates a secondary classifier (ResNet-18) into the YOLOv5m framework. This approach refines the predictions generated by the primary YOLOv5m model, thereby enhancing overall detection accuracy. The pipeline involves designing and implementing the primary detection model, preparing a custom dataset for the secondary classifier, designing and training the secondary classifier itself, and integrating this classifier into YOLO's detection pipeline. We used the vehicle coco dataset [8] to train the YOLOv5m model and prepared a dataset for the secondary classifier. The model architecture is shown in 1 [1].

3.1. Primary Detection Model

The initial stage of our detection pipeline utilizes the YOLOv5m model. It processes input images through a simple yet powerful convolutional neural network architecture to predict bounding boxes, class labels, and the respective confidence scores in a single detection. The pre-trained model is later fine-tuned to focus on four vehicle categories pertinent to our task: car, bus, motorcycle, and truck. Despite high performance, we observed that YOLOv5m occasionally produced false positives, misclassified non-vehicle regions as vehicles, and also detected the wrong class. To mitigate this issue, we introduced a secondary classification stage to validate and refine the model's predictions.

3.2. Dataset Preparation for Secondary Classifier

The efficacy of the secondary classifier depends on a well-prepared dataset that accurately represents the types of detections produced by YOLOv5m. For this, we constructed a

[1] YOLO v5 in this image is taken from [1]

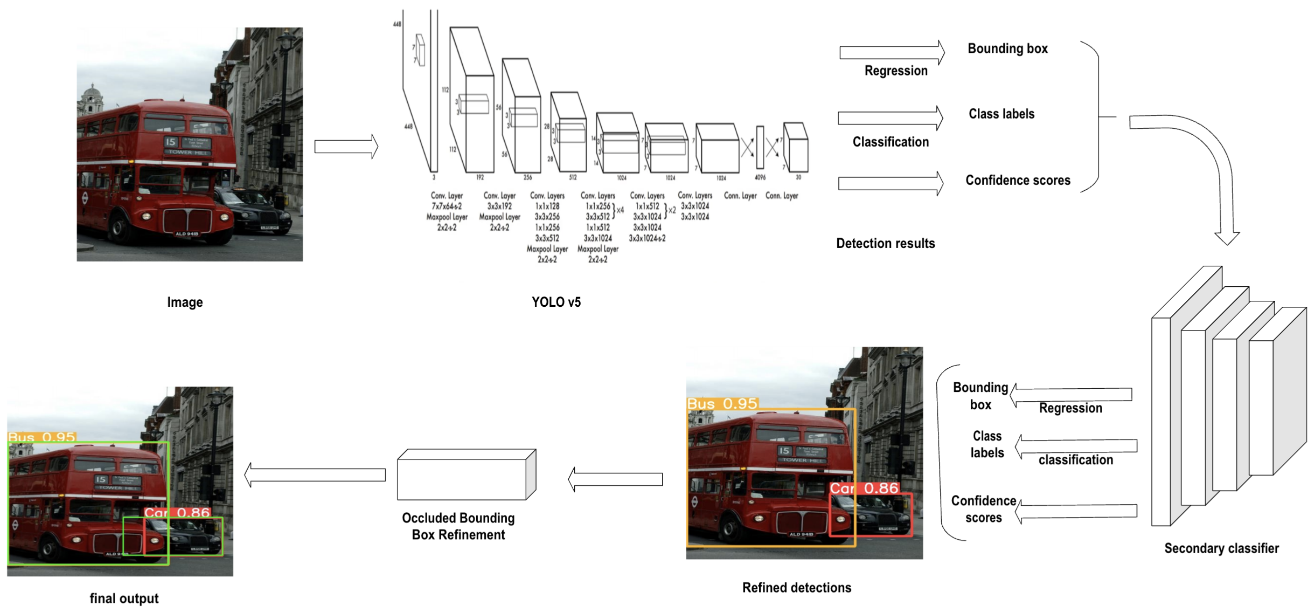


Figure 1. Overview of Object Detection Architecture

custom dataset to capture both true positives and false positives generated by the YOLOv5m model. Initially, we performed inference using YOLOv5m on the training images of the COCO vehicles data [8], which contained ground truth annotations for vehicles. We recorded the model's detections including predicted bounding boxes, class labels, and confidence scores for each image. For every detection, we extracted the corresponding region of interest (ROI) from the original image using the predicted bounding box coordinates. These ROIs were then resized to a uniform dimension of 224×224 pixels to match the input size required by the secondary classifier.

To label the detections accurately, we calculated the Intersection over Union (IoU) between each predicted bounding box and the ground truth bounding boxes. Detections with an IoU greater than 0.5 with a ground-truth box were labeled as *positive samples*, representing true positives. Conversely, detections with an IoU of 0.5 or less were labeled as *negative samples*, representing false positives. Thus, the resulting dataset consists of cropped ROIs organized into two classes: positive and negative. We used this dataset for training the secondary classifier to effectively distinguish between correct and incorrect detections produced by YOLOv5m.

3.3. Secondary Classifier Design and Training

The secondary classifier was designed to validate YOLOv5m's detections by filtering out false positives, thereby improving the overall precision of the detection pipeline. We selected the ResNet-18 architecture for its balance between computational efficiency and classification performance. The network's architecture was modified by replacing the original fully connected layer, which outputs logits for 1000 classes, with a new fully connected layer that outputs logits for two classes: positive and negative.

Data Augmentation and Preprocessing: To enhance the generalization capability of the secondary classifier, we applied several data augmentation techniques during training. These included random rotations to simulate different orientations, horizontal and vertical flips to account for reflections, scaling transformations to mimic size variations, and color jitter adjustments to address changes in lighting conditions. The augmented images were normalized using the mean and standard deviation values from the ImageNet dataset: a mean of $[0.485, 0.456, 0.406]$ and a standard deviation of $[0.229, 0.224, 0.225]$.

Training Procedure: The secondary classifier was trained using the cross-entropy loss function, which measures the discrepancy between the predicted class probabilities and the true labels. We used the Adam optimizer with an initial learning rate of 1×10^{-3} to update the model parameters. A cosine annealing scheduler was implemented to adjust the learning rate during training, gradually reducing it to prevent the optimizer from overshooting minima. The training was conducted over 25 epochs with a batch size of 64. To monitor the model's performance and prevent overfitting, we allocated 10% of the training data for validation purposes. We also applied regularization techniques such as dropout, with a rate of 0.5 before the final fully connected layer, and weight decay of 1×10^{-4}.

3.4. Integration of the Secondary Classifier into the Detection Pipeline

After training, we integrated the secondary classifier into the YOLO v5m detection pipeline as a post-processing module. The enhanced pipeline operates in several stages to ensure that only validated detections are retained.

An input image is first processed by the YOLOv5m model, which outputs a set of detections denoted as $\{(b_i, c_i, s_i)\}_{i=1}^{N}$, where b_i represents the bounding box coordinates, c_i the class label, and s_i the confidence score for the i-th detection. For each detection, the corresponding RoI is extracted from the original image using b_i. The RoI is resized to 224×224 pixels and passed through the secondary classifier, which outputs a probability distribution over the two classes: positive (p_{positive}) and negative (p_{negative}). Detections where $p_{\text{negative}} > 0.5$ are considered false positives and are discarded. The remaining detections, where $p_{\text{positive}} > 0.5$, are retained and constitute the refined set of detections. The refined detections $\{(b_i, c_i, s_i)\}$ are presented as the final output of the pipeline. This process effectively reduces false positives and enhances the overall precision of the object detection system.

3.5. Occluded Bounding Box Enhancements

Object detection models often struggle to accurately localize objects in cases of occlusion (when objects overlap or partially block each other). To address this, we implemented a post-processing method that adjusts predicted bounding boxes based on Intersection over Union (IoU) and proximity gaps. This approach enhances bounding box accuracy without retraining the model. We are enhancing object detection predictions by refining the bounding boxes to better account for occlusion scenarios. This process involves analyzing existing predictions (bounding boxes) and making adjustments based on their overlap and proximity to other objects in the image.

The success of this enhancement is closely tied to the prediction accuracy of the initial detection models. The goal of this enhancement is to generate more accurate bounding boxes and provide augmented data that can be used to retrain detection models for improved performance. This process aims to improve the model's ability to understand and handle occlusion scenarios more effectively. We observed that object detection—specifically for vehicles—tends to fail when the objects are partially occluded. This highlights the critical need for better predictions to ensure accurate localization and identification of occluded objects. By addressing these challenges, this enhancement provides a foundation for models to perform more robustly in complex environments.

Methodology: The process begins with the predicted images and their corresponding bounding box labels. YOLOv5m returns the bounding boxes in *xywh* format, that is, $\{$class$_{id}$, x_{center}, y_{center}, width, height, confidence score$\}$. To align it properly with our predictions, we convert them to Pixel format:

$$\text{Pixel Format: } (c, x_1, y_1, x_2, y_2)$$

$$x_1 = (x_{\text{center}} - \frac{w}{2}) \times \text{image\_width}$$

$$y_1 = (y_{\text{center}} - \frac{h}{2}) \times \text{image\_height}$$

$$x_2 = (x_{\text{center}} + \frac{w}{2}) \times \text{image\_width}$$

$$y_2 = (y_{\text{center}} + \frac{h}{2}) \times \text{image\_height}$$

The process begins with the model's predictions, where bounding boxes with low confidence scores (< 0.25) are flagged as potential candidates for occlusion. A lower confidence score is typically indicative of uncertainty in detection, often caused by partial visibility of objects. Next, the IoU between each bounding box and other bounding boxes in the same image is calculated. IoU measures the overlap between two boxes, and a threshold is applied to identify overlapping or occluded objects. If the IoU exceeds this threshold, the bounding boxes are flagged for enhancement. This ensures that even minor overlaps are accounted for, as they could indicate occlusion.

The flagged bounding boxes are dynamically expanded based on their IoU, relative height, and width. The adjustments are made proportionally, ensuring that expansions align with the direction of overlap or proximity. For example, if the IoU indicates occlusion from the right side, the box expands to the right. Similarly, vertical adjustments are applied if the occlusion occurs above or below the object. The degree of expansion is determined by a function of IoU and the bounding box's relative dimensions. Larger overlaps result in more significant adjustments, while small overlaps result in minimal expansion. This ensures that the process is adaptive and context-aware. All expansions are constrained within the image boundaries to maintain realistic adjustments.

By combining confidence scores, IoU, and dynamic dimension scaling, this method systematically enhances bounding boxes to better account for occlusions. The refined bounding boxes can then be used to retrain the model, enabling it to learn and handle occlusion scenarios more effectively. To enhance bounding box refinement dynamically, a regression model can be trained to predict adjustments (dx_1, dy_1, dx_2, dy_2) based on input features such as IoU, confidence scores, bounding box dimensions, and class labels. This model learns patterns from training data where adjustments are annotated or simulated, enabling it to generalize to diverse occlusion scenarios.

3.6. Implementation Details

The models were implemented using the PyTorch deep learning framework. All experiments were conducted on Google Colab Pro with an NVIDIA L4 GPU with 53 GB of RAM. We used the variant of YOLOv5 (YOLOv5m) for its balance between speed and accuracy. We fine-tuned the model on our vehicles COCO dataset [8] for 10 epochs. The ResNet-18 model was initialized with weights pre-trained on ImageNet. Initially, only the final layer was retrained to adapt to our two-class problem. Subsequently, we fine-tuned the entire network to improve performance. We conducted hyperparameter tuning to determine the optimal learning rate and batch size, with the best results achieved using a learning rate of 1×10^{-3} and a batch size of 64. We also applied regularization techniques, including dropout and weight decay to prevent overfitting. The code with the trained models and datasets are publicly available in https://github.com/amit-sarker/vehicle-detection-with-occlusion.

3.7. Evaluation Metrics

To assess the performance of the enhanced detection pipeline, we employed the COCO evaluation metrics, which provide a comprehensive analysis of object detection models. We calculated the **Average Precision (AP)** at IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05. This metric evaluates the precision of the model across different levels of localization accuracy. The **Average Recall (AR)** was evaluated across various object sizes—small, medium, and large—to determine the model's ability to detect objects of different scales.

4. Results

Dataset: We conducted the evaluation of our proposed two-stage detection pipeline using a dataset [8] comprising 18998 images annotated for object detection tasks. The dataset is divided into three subsets: a training set with 13300 images (70% of the total dataset), a validation set containing 3798 images (20%), and a test set of 1900 images (10%). The images in the dataset featured diverse scenes and conditions, encompassing urban and rural environments, different weather conditions, and varying times of day. The object categories in the annotations are **Bus**, **Car**, **Motorcycle**, and **Truck**, with bounding boxes marking the locations of these vehicles in each image.

4.1. Quantitative analysis

Baselines: To evaluate the effectiveness of our proposed detection pipeline (YOLOv5m + ResNet), we compared its performance against several baseline models from the YOLOv5 family: **YOLOv5n**, **YOLOv5s**, **YOLOv5m**,

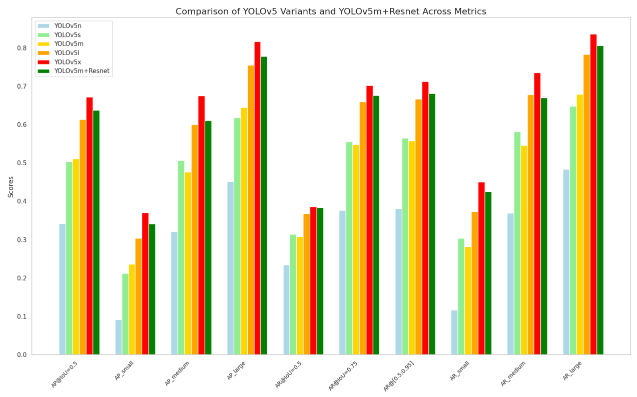


Figure 2. Performance evaluation of YOLOv5m + ResNet with other YOLOv5 variants across all classes.

YOLOv5l, and **YOLOv5x**. These models represent different scales within the YOLOv5 architecture and vary in network depth and width to balance between computational efficiency and detection precision. YOLOv5n (nano) and YOLOv5s (small) are lightweight models designed for speed, while YOLOv5l (large) and YOLOv5x (extra-large) offer higher accuracy at the cost of increased computational resources. YOLOv5m (medium) serves as a middle ground between these extremes.

We selected these baselines to provide a comprehensive evaluation across a spectrum of model complexities. By comparing our proposed model (YOLOv5m + ResNet), against these baselines, we aimed to demonstrate the efficacy of our approach in improving detection performance without incurring significant computational overhead. We chose YOLOv5m as the base for our model because it offers a balance of speed and accuracy suitable for real-time applications. That makes it an ideal candidate for enhancement through our secondary classification stage. For a fair comparison, we trained all the baselines for 10 epochs using our vehicle coco dataset.

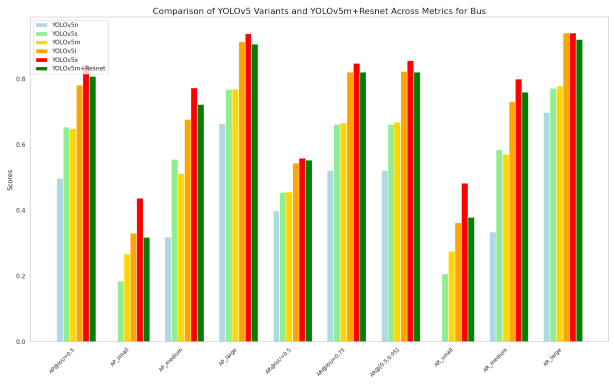
4.1.1 Performance Analysis (All Classes)

We assessed the performance of each model using standard object detection metrics, focusing on the collective results across all classes: **Bus**, **Car**, **Motorcycle**, and **Truck**. The metrics include Average Precision (AP) at IoU thresholds, specifically AP@IoU=0.5, as well as AP for small, medium, and large objects. Additionally, we evaluated Average Recall (AR) under various conditions to gauge the models' abilities to detect objects of different sizes and at different IoU thresholds. The evaluation results are shown in Figure 2 and metric scores can be found in Table 1.

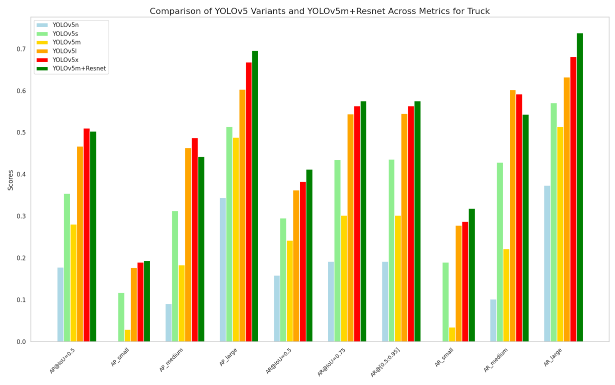
From the results, we observe that our proposed model (YOLOv5m + ResNet), achieved an AP@IoU=0.5 of 0.637, surpassing the baseline YOLOv5m model's AP of 0.511 by

Table 1. Performance metrics for different YOLOv5 models and the proposed YOLOv5m + ResNet model.

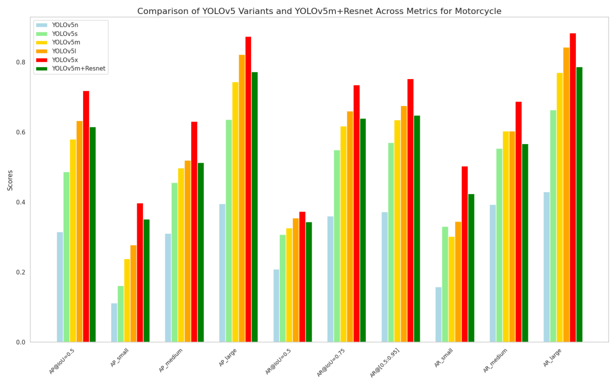
Metric	**YOLOv5n**	**YOLOv5s**	**YOLOv5m**	**YOLOv5l**	**YOLOv5x**	**YOLOv5m + ResNet**
AP@IoU=0.5	0.342	0.504	0.511	0.613	0.672	**0.637**
AP_small	0.092	0.212	0.236	0.304	0.370	**0.341**
AP_medium	0.322	0.506	0.477	0.600	0.675	**0.611**
AP_large	0.451	0.617	0.645	0.754	0.816	**0.777**
AR@IoU=0.5	0.234	0.315	0.308	0.369	0.386	**0.384**
AR@IoU=0.75	0.376	0.555	0.548	0.659	0.702	**0.676**
AR@[0.5:0.95]	0.381	0.565	0.557	0.667	0.712	**0.681**
AR_small	0.117	0.304	0.282	0.374	0.450	**0.425**
AR_medium	0.370	0.582	0.547	0.678	0.735	**0.670**
AR_large	0.484	0.648	0.679	0.783	0.836	**0.806**



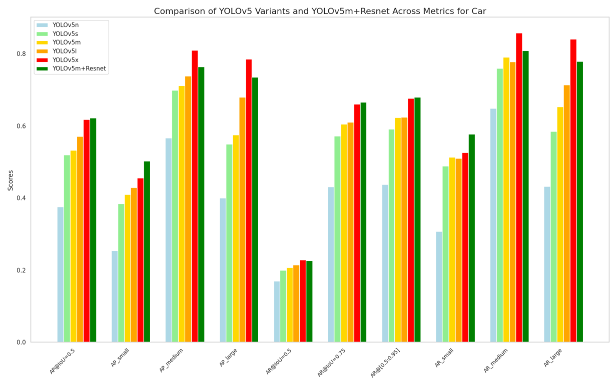
(a) Performance evaluation for **Bus**.



(b) Performance evaluation for **Truck**.



(c) Performance evaluation for **Motorcycle**.



(d) Performance evaluation for **Car**.

Figure 3. Comparison of YOLOv5m + ResNet with other YOLOv5 variants across all vehicle categories.

a significant margin. This improvement highlights the effectiveness of integrating the ResNet-18 secondary classifier in enhancing detection accuracy. When examining object sizes, our model demonstrated consistent performance gains across all categories:

- AP_small increased from 0.236 (YOLOv5m) to 0.341. This indicates a substantial enhancement in detecting small vehicles. Small objects are challenging due to their limited pixel information.
- AP_medium improved from 0.477 to 0.611. This showcases better precision in identifying medium-sized vehicles.
- AP_large increased from 0.645 to 0.777. This reflects the model's strengthened capability in accurately detecting large vehicles.

The Average Recall metrics also showed notable improvements:

- AR@IoU=0.75 increased from 0.548 to 0.676. This suggests that our model is more effective at higher IoU thresholds, which require more precise localization.

- AR@[0.5:0.95] increased from 0.557 to 0.681. This demonstrates enhanced recall over a range of IoU thresholds.
- Improvements in AR_small, AR_medium, and AR_large further confirm the model's better detection capabilities across different object sizes.

Comparing our model to larger YOLOv5 variants like YOLOv5l and YOLOv5x, we notice that YOLOv5x achieved a higher AP@IoU=0.5 of 0.672. However, this comes at the cost of increased computational resources and longer inference times due to the model's larger size. Our proposed model offers a competitive AP@IoU=0.5 while maintaining the efficiency associated with YOLO v5m. This makes it more suitable for real-time applications where resource constraints are a consideration.

4.1.2 Per-Class Performance Analysis

In addition to the overall performance, we conducted a detailed analysis of the detection results for each object class: **Bus**, **Truck**, **Motorcycle**, and **Car**. The performance metrics for these classes were evaluated using the same AP and AR metrics as before, focusing on the impact of our proposed model on individual object categories.

Bus: For the Bus category, our proposed model achieved an $AP@IoU_{0.5}$ of 0.807, a substantial improvement over the baseline YOLOv5m's AP of 0.649. This performance is close to that of the larger YOLOv5x model, which achieved an AP of 0.841. The AP for small buses increased from 0.268 (YOLOv5m) to 0.318 with our model. This indicates enhanced detection of small-sized buses. For medium and large buses, the AP improved from 0.512 and 0.770 to 0.722 and 0.906, respectively. The AR metrics showed similar trends, with AR@IoU=0.75 increasing from 0.667 to 0.821, demonstrating better localization accuracy. Figure 3a shows the evaluation results for the Bus class.

Truck: In the case of Truck detection, our model achieved an AP@IoU=0.5 of 0.503. This significantly outperforms the baseline YOLOv5m's AP of 0.282. This result is comparable to the YOLOv5x model, which achieved an AP of 0.511. The AP for small trucks increased from 0.030 (YOLOv5m) to 0.194 with our model. This indicates a substantial enhancement in detecting small trucks. For medium and large trucks, the AP improved from 0.184 and 0.489 to 0.443 and 0.696, respectively. The AR@IoU=0.75 increased from 0.303 to 0.576, reflecting improved precision in truck localization. Figure 3b shows the evaluation results for the Truck class.

Motorcycle: For the Motorcycle class, our proposed model achieved an AP@IoU=0.5 of 0.616. This surpasses the baseline YOLOv5m's AP of 0.581. While this performance is slightly lower than that of YOLOv5x (0.719), our model demonstrated significant improvements in detecting small motorcycles, with the AP_small increasing from 0.239 (YOLOv5m) to 0.351. The AP for medium motorcycles improved from 0.450 to 0.514 and for large motorcycles from 0.745 to 0.773. The AR metrics indicate enhanced recall across all object sizes, with AR@[0.5:0.95] increasing from 0.635 to 0.649. Figure 3c shows the evaluation results for the Motorcycle class.

Car: In the Car category, our model achieved an AP@IoU=0.5 of 0.622, outperforming the baseline YOLOv5m's AP of 0.532 and marginally exceeding YOLOv5x's AP of 0.618. The AP_small for cars increased from 0.410 (YOLOv5m) to 0.503. This indicates a notable improvement in detecting small cars. For medium-sized cars, the AP improved from 0.712 to 0.764, while the AP for large cars increased from 0.576 to 0.735. The AR@IoU=0.75 saw an increase from 0.606 to 0.666, reflecting better localization precision. Figure 3d shows the evaluation results for the Car class.

4.2. Qualitative analysis

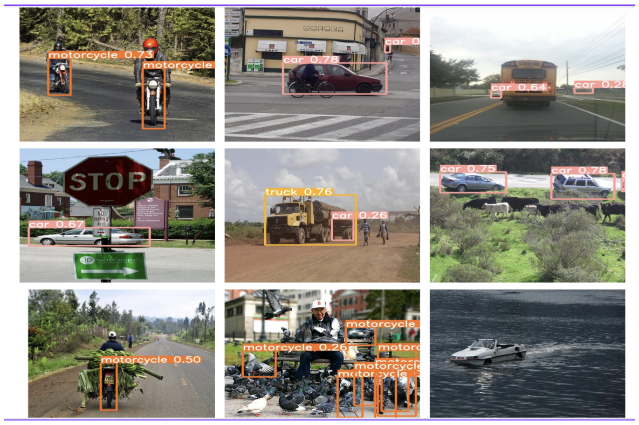
We, further, verify the quality of detections produced by the model. As shown in Figure 5, our proposed module readjusts the bounding boxes of objects that are partly missing from the image, while the detections of unoccluded vehicles remain consistent. This refinement indicates that the module calculates the appropriate bounding box dimensions as if the object in its front is missing. Also, it can be noticed from sthe econd image in Figure 5 that the model performs better than baselines even with multiple vehicles stacked. However, the confidence scores and vehicle classes are hardly improved with this module.

From Figure 4, we observe that the ResNet backbone assists YOLOv5m in finding some of the missed detections, and improves the confidence levels of the correct detections while removing the incorrect predicted bounding boxes.

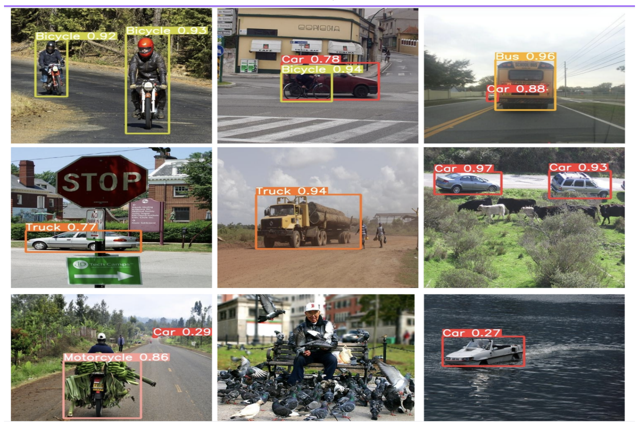
Though our proposed model performs well, it degrades the performance of YOLOv5m to a small extent by its inability to detect far-away objects. For instance, Figure 6 shows the first row, where the backbone misses some objects barely visible in the scene. It can not detect objects in less illumination, as seen in the second row. When multiple vehicles of similar colors are stacked one after the other as shown in third row of 6, the model finds it challenging. This could be attributed to the model's performance being heavily dependent on initial detections.

5. Conclusion

In this work, we presented a two-stage detection pipeline that enhances vehicle detection accuracy by integrating a ResNet-18 secondary classifier into the YOLOv5m framework and implemented occlusion-aware bounding-box en-

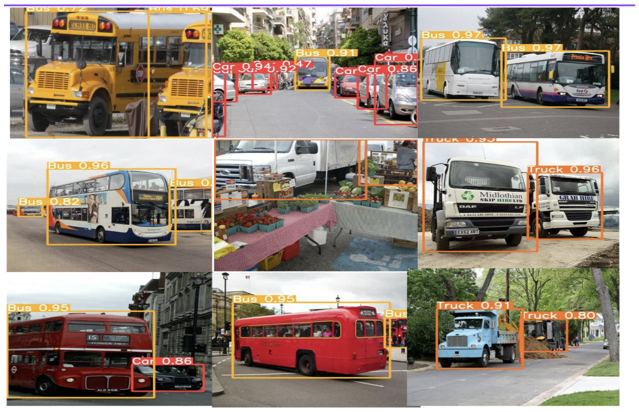


(a) Detections by YOLOv5m

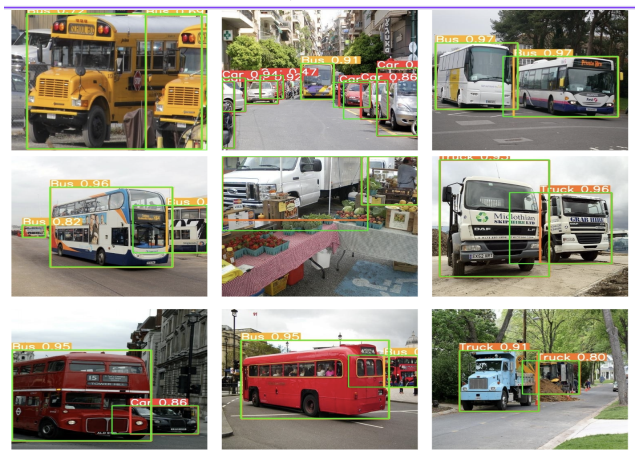


(b) Detections by YOLOv5m + ResNet

Figure 4. Comparisons of detections with and without the secondary classifier (ResNet).



(a) Detections by YOLOv5m + ResNet



(b) Detections by YOLOv5m + ResNet with Occlusion-Aware Module

Figure 5. Comparisons of detections with and without the occlusion-aware module.

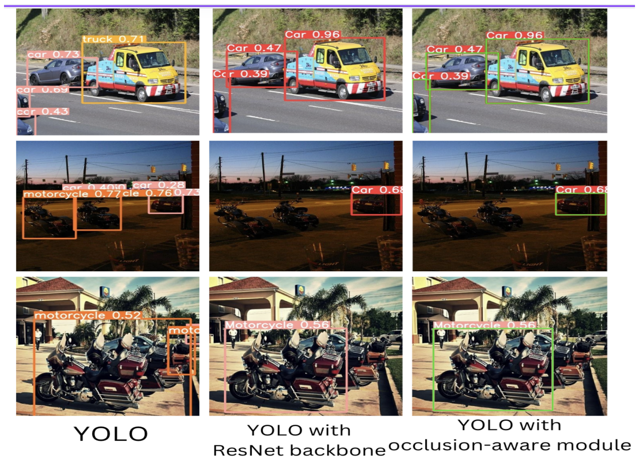


Figure 6. Error analysis of a few poor detections by models.

hancements. We show that IoU with earlier predictions and aspect ratio serve as important information for these enhancements. Our approach effectively reduces false positives and misclassification, and improves localization accuracy in occluded scenarios without significant computational overhead. Experimental results demonstrate significant improvements in Average Precision (AP) and Average Recall (AR) across all vehicle categories, especially for small and medium-sized objects. The proposed method offers a practical solution for real-time applications in autonomous driving and ADAS, contributing to safer and more reliable object detection systems in complex environments. We conclude that while IoU and aspect ratio are important, they are not sufficient for predicting object dimensions when occluded.

References

[1] Yolo: Algorithm for object detection explained, 2023. 2

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE*

transactions on pattern analysis and machine intelligence, 43(5):1483–1498, 2019. 2

[3] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. Transmot: Spatial-temporal graph transformer for multiple object tracking. In *Proceedings of the IEEE/CVF Winter Conference on applications of computer vision*, pages 4870–4880, 2023. 2

[4] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. 2

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2

[6] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6054–6063, 2019. 2

[7] Liting Lin, Heng Fan, Zhipeng Zhang, Yong Xu, and Haibin Ling. Swintrack: A simple and strong baseline for transformer tracking. *Advances in Neural Information Processing Systems*, 35:16743–16754, 2022. 2

[8] Vehicle MSCOCO. Vehicles-coco dataset. `https://universe.roboflow.com/vehicle-mscoco/vehicles-coco`, 2022. visited on 2024-10-31. 1, 2, 3, 5

[9] J Redmon. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 1, 2

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 2

[11] Yining Shi, Jingyan Shen, Yifan Sun, Yunlong Wang, Jiaxin Li, Shiqi Sun, Kun Jiang, and Diange Yang. Srcn3d: Sparse r-cnn 3d for compact convolutional multi-view 3d object detection and tracking. *arXiv preprint arXiv:2206.14451*, 2022. 2

[12] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14454–14463, 2021. 2

[13] Jian Wei, Jianhua He, Yi Zhou, Kai Chen, Zuoyin Tang, and Zhiliang Xiong. Enhanced object detection with deep convolutional neural networks for advanced driving assistance. *IEEE transactions on intelligent transportation systems*, 21 (4):1572–1583, 2019. 1

[14] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International journal of computer vision*, 129:3069–3087, 2021. 2

[15] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*, pages 1–21. Springer, 2022. 2

[16] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2