Team Passphrase: Dhaka AI Vehicle Detection

Sowmen Das*, Nazia Tasnim*, Md. Istiak Hossain Shihab*, Mohammad Faiyaz Khan*, and Rafid Ameer Mahmud†

*Department of Computer Science and Engineering, Shahjalal University of Science and Technology
†Department of Computer Science and Engineering, University of Dhaka.

Objective

Traffic detection comes with many different challenges depending on the specific characteristics of the problem domain. Dhaka traffic has its own distinctive elements and dynamics. To design an effective object detection model that captures the essence of dhaka traffic, careful model optimization and data engineering is needed. So, we focus on developing a robust pipeline using an ensemble of existing SOTA models, to ensure improved generalization and less dependency on model complexity.

Dataset Exploration & Preparation

A major challenge in improving detection performance was the skewed distribution of training data, as shown in Fig. 1. The provided data was insufficient for training existing models as they ovefit very quickly. Moreover, a large portion of the data had faulty labels and corrupted metadata. To fix these issues we employed elaborate data engineering. Firstly, we removed the corrupted samples and resized all images to 1024 × 1024. Secondly, we used LabelImg [6] to clean and fix faulty annotations and relabel the images. Finally, to deal with the scarcity of training data, we added images from PoribohonBD [5] to our train set that correlated with the existing classes. The resulting class distribution after data cleaning is shown in Fig. 1. For effective model selection, we used a 3-fold Stratified Cross Validation on the final dataset.

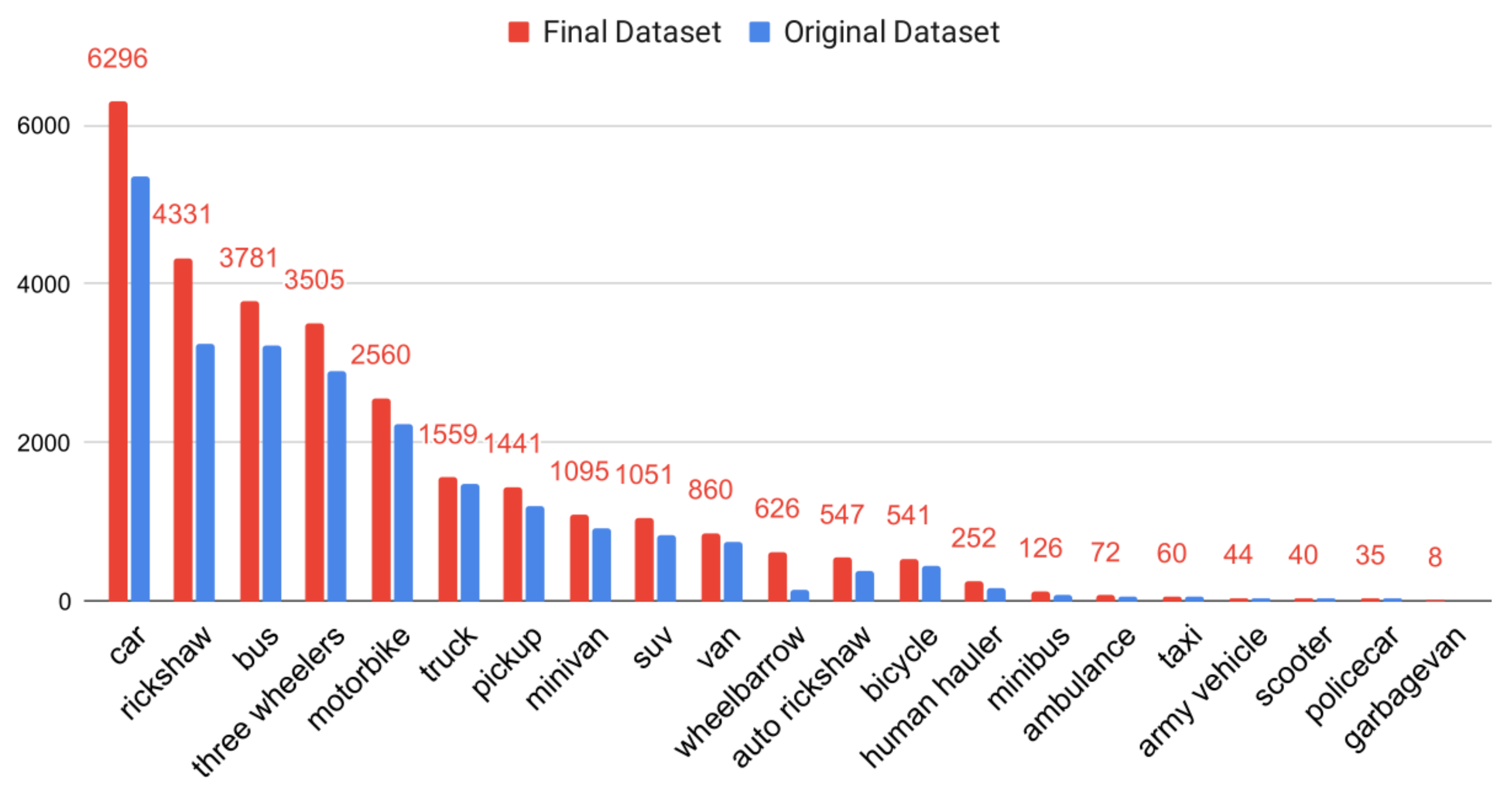

Fig. 1: Distribution of samples for each class before and after data cleaning and addition of external data.

To improve data variation and generalization we relied on heavy train data augmentation. We trained all our models using transfer learning to deal with the data scarcity. We found a great performance boost using Cut-mix, Mixup and Mosaic augmentation along with color corrections and random geometric transforms. Samples of train images are shown in Fig. 2.

Fig. 2: Train image samples demonstrating various data augmentations.

Workflow Optimization

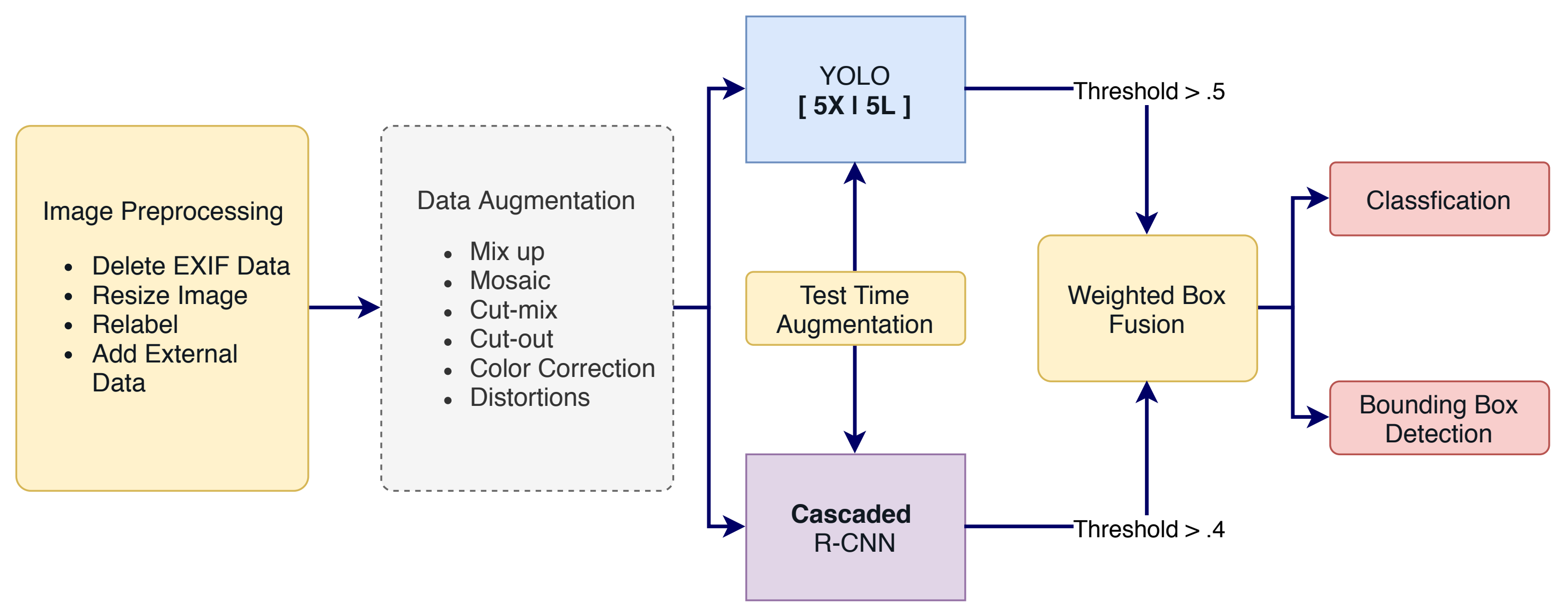

Fig. 3: Addition of Data.

Model Selection: We evaluated a number of SOTA architectures on the data with transfer learning. From our experiments as shown in Table 1, we see that individually YOLOv5X [2] has the best mAP followed by Cascaded-RCNN [1]. We decided on using an weighted ensemble of these two architectures for our final pipeline. The overall training and inference pipeline is shown in Fig. 3.

Model	Specification	Val mAP
EfficientDet	Models D0-D7x	19.05%
Faster RCNN-32x8d	32 groups and group width 8	44.71%
Faster RCNN-64x4d	64 groups and group width 4	42.21%
Panoptic FPN - ResNet101	Panoptic Segmentation	39.23%
Cascaded-RCNN - ResNet-50	Cascaded Feature Pyramid (RFP)	65.6%
Yolov5-L	One-shot detector	72.4%
Yolov5-X	-	75.6%
Cas-RCNN + Faster-RCNN	WBF Ensemble	62.7%
Cas-RCNN + Yolov5X	WBF Ensemble	77.8%
Cas-RCNN + Faster-RCNN + Yolov5X	WBF Ensemble	75.64%
Yolov5L + Yolov5X	NMS ensemble + TTA	87.4%
Yolov5L + Yolov5X + Cas-RCNN	WBF Ensemble + TTA	**89.5**%

Tab. 1: Results of different experiments for model evaluation.

Optimization & Fine-Tuning: For model tuning and optimization we tuned the hyper-parameters using genetic evolution algorithm [3] for 300 generations against an optimization function. Moreover for final inference we used Test Time Augmentation (TTA) which resulted in an additional boost in performance. An important part of model tuning was determining the correct confidence threshold. Since we had a very limited number of submissions, we replicated the leaderboard evaluation metric using a partial self labelled test dataset. We tuned the NMS, WBF, and confidence thresholds using this metric on the test set. This was critical in improving our final leaderboard score as we found that tuning the thresholds resulted in a huge boost in test performance. The final hyper-parameters are listed in Table. 2. All our models were trained on Colab-Pro instances using V100 GPU's.

Model	Learning Rate	Optimizer	Train Epoch	Loss Function	Scheduler
Cascaded R-CNN	.0025	SGD	24	Focal Loss	Linear
Yolo	0.00056	ADAM	300	Focal Loss	Cyclic

Tab. 2: Final hyper-prameter values.

Ensembling: To perform ensembling on the predictions of multiple models, we used Weighted Box Fusion (WBF)[4] approach which utilizes confidence scores of all proposed bounding boxes to construct the mean prediction box. The predictions from the YOLO models were thresholded with a 55% confidence score, while a 40% threshold was used for Cascaded-RCNN. These outputs were passed to WBF algorithm, with a weight of 2 and 1, respectively.

Results

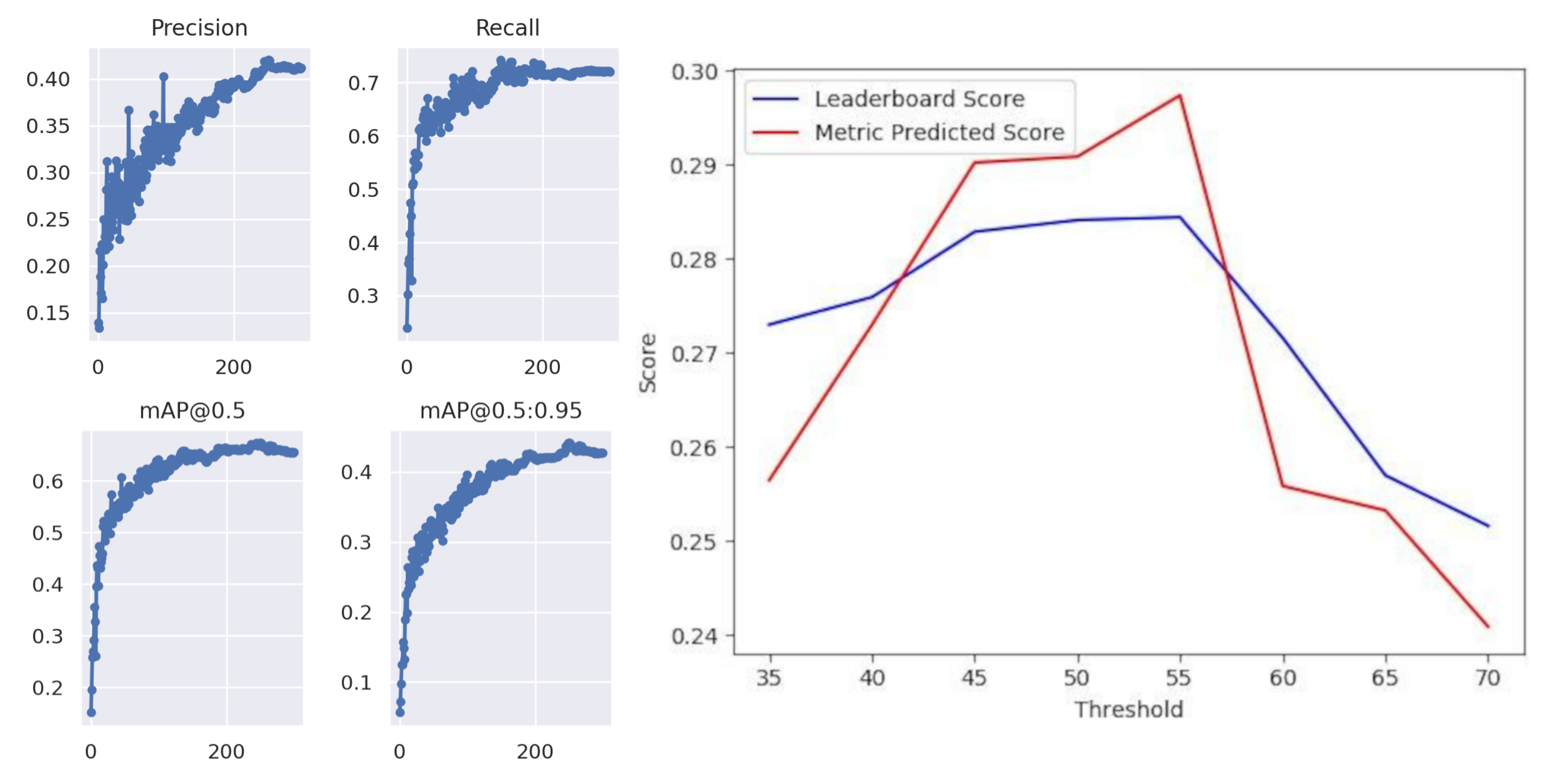

Fig. 4: Comparison of score, Precision-Recall and mAP.

Fig. 4 (right) shows the predicted and actual leaderboard scores for different confidence thresholds, and (left) the Precision-Recall and mAP of the Yolov5X on the train set.

Conclusion

The focus of our solution was to ensure simplicity and modularity. We incorporated transfer learning and used pretrained state of the art backbone networks to extract fundamental object features. Using transfer learning implies that our model can be easily applied to a wide range of traffic detection problems. After evaluating and testing various object detection algorithms and architectures, we compiled the best performing ones and achieved better performance by efficiently combining them. In the end, we used an emulated metric to find the best combination of hyperparameters. All these techniques solidified our position in phase 1 and boosted it in phase 2.

References

[1] Zhaowei Cai and Nuno Vasconcelos. "Cascade R-CNN: Delving into High Quality Object Detection". In: *CoRR* abs/1712.00726 (2017). arXiv: 1712.00726. URL: http://arxiv.org/abs/1712.00726.

[2] Glenn Jocher et al. *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements.* 2020. DOI: 10.5281/zenodo.4154370. URL: https://doi.org/10.5281/zenodo.4154370.

[3] Michael Orlov, Moshe Sipper, and Ami Hauptman. "Genetic and Evolutionary Algorithms and Programming: General Introduction and Application to Game Playing". In: 2009. ISBN: 978-0-387-30440-3. DOI: 10.1007/978-0-387-30440-3_243. URL: https://doi.org/10.1007/978-0-387-30440-3_243.

[4] Roman Solovyev and Weimin Wang. "Weighted Boxes Fusion: ensembling boxes for object detection models". In: *CoRR* abs/1910.13302 (2019). arXiv: 1910.13302. URL: http://arxiv.org/abs/1910.13302.

[5] Shaira Tabassum et al. "Poribohon-BD: Bangladeshi local vehicle image dataset with annotation for classification". In: *Data in Brief* 33 (2020), p. 106465. ISSN: 2352-3409. DOI: https://doi.