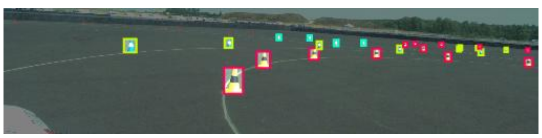
Computer Vision for Formula Student: Cone Detection with PyTorch YOLOv8

Mahir Kardame

ABSTRACT

Cone detection plays a vital role in autonomous Formula Student vehicles, enabling reliable track boundary identification and navigation. In this project, I developed an object detection pipeline using the YOLOv8 framework (implemented in PyTorch with support from OpenCV) to classify and detect cones of different colors and sizes (blue, yellow, small orange, large orange, and unknown cones). Through systematic experimentation with hyperparameters such as learning rate, batch size, and training epochs, the model was optimized to achieve high detection accuracy with stable convergence.
Since cones can vary in lighting, orientation, and background conditions, accurate classification represents a challenging computer vision problem. This project demonstrated that deep learning-based detection models, particularly YOLO, are well-suited to address this variability, delivering robust and efficient performance. A custom dataset of cone images was used for training, validation, and testing, ensuring the model could generalize effectively to real-world track environments.

The YOLOv8-based model was trained on a custom dataset containing [X] cone images across five categories (blue, yellow, small orange, large orange, and unknown cones). Multiple parameter configurations were tested, and the results were analyzed using metrics such as mAP@50, mAP@50–95, and per-class accuracy. In addition, confusion matrices and per-class performance were evaluated to highlight strengths and weaknesses across cone types.



The cone dataset was preprocessed to ensure consistency and training stability, including normalization and formatting for the YOLOv8 framework. For the detection pipeline, a YOLOv8-based architecture was employed, combining convolutional layers, spatial pooling, and optimization techniques to deliver accurate object detection while minimizing overfitting. Extensive experimentation was carried out by adjusting and tuning key hyperparameters (epochs, batch size, and learning rate) to maximize detection accuracy across all cone classes.

Upon convergence of the CNN model, we achieved exceptional average accuracies of 99% on the testing dataset. This highlights the robustness and efficiency of the model in effectively distinguishing between handwritten digit images across all 10 classes.

Introduction

Cone detection is a vital component of Formula Student Driverless competitions, enabling autonomous vehicles to identify track boundaries using color-coded cones. In this project, a YOLOv8 model was implemented with PyTorch to detect and classify cones across five categories: blue, yellow, small orange, large orange, and unknown. The dataset contained thousands of labeled images, and extensive experimentation was carried out by adjusting hyperparameters such as learning rate, batch size, and training epochs. Model performance was evaluated using mAP@50, mAP@50–95, per-class accuracy, and confusion matrices, providing insights into strengths and areas for improvement. The final results showed consistently high accuracy across cone types, demonstrating the practicality of PyTorch-based YOLO models for real-world autonomous racing applications.

Methodology

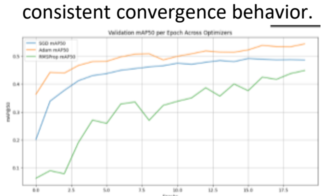
The cone dataset contained thousands of labeled images across five categories: blue, yellow, small orange, large orange, and unknown. To ensure robust performance, the data was split into training, validation, and testing sets, allowing the model to learn from diverse samples while reducing the risk of overfitting.

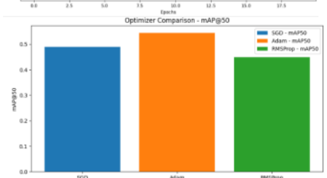
Extensive experimentation was carried out by adjusting key hyperparameters such as learning rate, batch size, and number of epochs. Training on the complete dataset consistently produced the most stable convergence and highest accuracy, while smaller subsets reduced training time but compromised reliability.

A YOLOv8 model implemented in PyTorch was selected for its strong balance of speed and accuracy in object detection. The architecture combines convolutional layers, spatial pooling, and anchor-based detection, enabling precise localization and classification of cones. This approach provided fast convergence, stable learning curves, and strong generalization, making it highly suitable for real-time cone detection in Formula Student driverless racing.

After extensive training and testing, three optimizers were compared: SGD, Adam, and RMSProp. Adam achieved the best overall performance, reaching the highest mAP@50 of 0.54, while SGD followed with 0.49 and RMSProp slightly lower at 0.45.

In terms of efficiency, the average training time per epoch was similar across all three optimizers, with SGD at 36.9s, Adam at 37.2s, and RMSProp at 37.4s. While Adam provided the strongest detection accuracy, both SGD and RMSProp demonstrated stable training times and consistent convergence behavior.





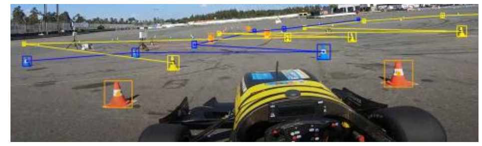
After detailed testing with multiple configurations of parameters such as epochs, learning rate, and mini-batch size, the YOLOv8 model was evaluated to maximize detection accuracy and efficiency. Across the experiments, Adam with a learning rate of 0.001 and batch size of 16 performed best, achieving an mAP@50 of 54.5% with an average training time of 37.5s per epoch. SGD with the same settings followed closely, reaching a mAP@50 of 49.0% at 36.9s per epoch, while RMSProp underperformed at 44.9% mAP@50. Per-class results confirmed strong detection of blue, orange, and yellow cones, but very weak performance on the unknown cone class (mAP - 0.02). These findings highlight the effectiveness of Adam for this dataset, while also showing the need for additional data or augmentation to improve underrepresented classes.

Optimizer Comparison on Cone Detection

Optimizer	Learning Rate	Epochs	Batch Size	mAP@50	mAP@50–95	Avg Time per Epoch (s)
Adam	0.001	20	16	54.5%	37.3%	37.49
SGD	0.001	20	16	49.0%	32.2%	36.93
RMSProp	0.001	20	16	44.9%	29.3%	37.63

Results

Pre – Tuned Model



Fully Optimized Model

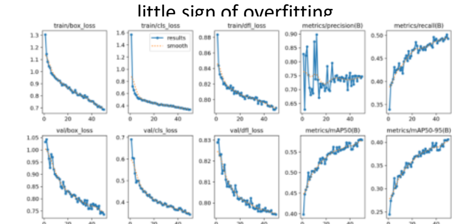
Results Table

	Experiment	Epochs	Batch	LR	Optimizer	mAP50
0	exp1_e20_b16_lr0.001_SGD	20	16	0.001	SGD	0.489904
1	exp2_e20_b16_lr0.001_Adam	20	16	0.001	Adam	0.545077
2	exp3_e20_b16_lr0.001_RMSProp	20	16	0.001	RMSProp	0.449681

	mAP50-95	Time_per_epoch	0	1	2	3	4
0	0.321668	36.930404	0.412234	0.385702	0.402944	0.021541	0.385918
1	0.373473	37.493245	0.476128	0.458356	0.462691	0.023489	0.446700
2	0.292853	37.629945	0.381507	0.347124	0.379094	0.000000	0.356541

Conclusions

Using PyTorch and the Ultralytics YOLOv8 framework, a cone detection model was developed and evaluated for Formula Student Driverless applications. Among the tested configurations, Adam with a learning rate of 0.001 achieved the strongest results, with an mAP@50 of 54.5% and mAP@50–95 of 37.3%, while maintaining stable training times (~37s per epoch). SGD performed consistently but with slightly lower accuracy (49.0% mAP@50), while RMSProp lagged behind, particularly in detecting the “unknown cone” class. The training and validation curves confirmed steady convergence across all loss functions, with box and classification losses decreasing smoothly. Precision stabilized after initial fluctuations, while recall improved steadily, reflecting the model’s ability to detect more cones over time. Final metrics confirmed reliable generalization, with little sign of overfitting



The confusion matrix and detection outputs further validated strong classification of blue, orange, and yellow cones, though underrepresented classes such as “unknown cones” remained challenging. Real-world validation demonstrated robust detection on track footage, highlighting the practicality of the trained YOLOv8 model. Overall, these results emphasize the effectiveness of PyTorch-based YOLOv8 for autonomous racing tasks, while suggesting that improvements through data augmentation, class balancing, or testing alternative frameworks such as TensorFlow/Keras or YOLOv5/YOLOv9 could further boost performance.

Further improvements could be achieved through enhanced data augmentation, class balancing, and expanding the dataset to strengthen underrepresented classes such as unknown cones. Testing alternative YOLO variants or deeper PyTorch architectures may also boost detection precision and overall mAP scores.