Alma Mater Studiorum – Università di Bologna

School of Engineering and Architecture

Master's Degree in Artificial Intelligence

Machine Learning for Computer Vision (ML4CV)

3 CFU Project

Improving Real-Time Cone Detection for Autonomous Formula SAE Racing with Modern YOLO Architectures

Student:

Nicolas Cridlig

Supervisor:

Prof. Samuele Salti

Academic Year 2024–2025

January 2026

Abstract

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context: Formula Student Driverless

This report addresses an element of the work undertaken by Unibo Motorsport (UBM) to succeed in the Formula SAE (FSAE) competition. Since 2017, the competition requires cars to autonomously navigate an unknown track delineated by colored traffic cones at maximum speed. To do so, the software and hardware stack must be efficient, reliable, and well integrated. The perception of the colored cones is vital due to the garbage-in, garbage-out moniker: even the best written code cannot produce good results on bad data. Therefore, in this report we increase successful cone detections by utilizing the newest generation of YOLO models alongside a new dataset. This research is vital, last summer, UBM competed at Formula Student Germany, and this year we will compete again from the 11-16th of August 2026.

1.2 Problem Statement

For perception, the software stack relies upon a custom trained YOLO object detector to identify track cones from a stereocamera so that they may be reconstructed with a multi-stage matching algorithm. The baseline model, YOLOv11, has room for improvement. It is trained on base hyperparameters and any tuning or analysis was lost. The mAP_{50}=0.666 and while this figure is our quantitative benchmark, qualitative issues registered by the team include exposure sensitivity and false positives. The model is constrained by both accuracy and real time speed. The stereocamera operates at 60fps which allows for a maximum control loop time of 16.7 ms. Any time perception takes out of this hard limit, is less time for the decision and control algorithms to operate.

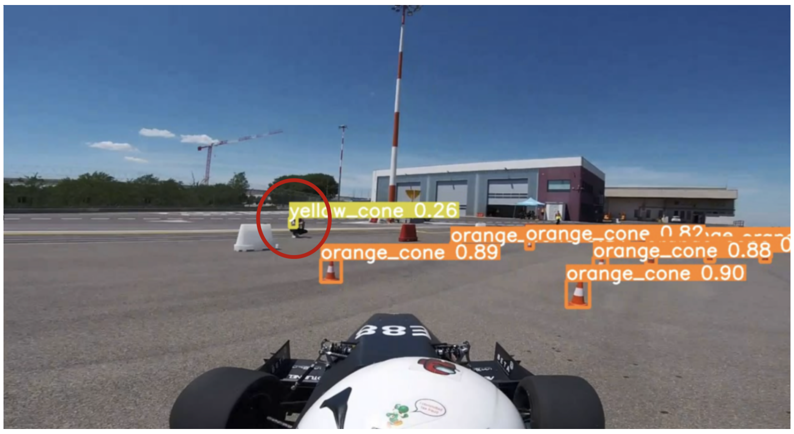

Figure 1.1: Example false positive: a person misclassified as a yellow cone by the baseline YOLOv11n detector.

1.3 Objectives

The project objectives are:

1. Reproduce and establish a reproducible baseline
2. Evaluate modern YOLO architectures (YOLO12n, YOLO26n)
3. Explore two-stage training (pre-training on larger dataset + fine-tuning)
4. Assess hyperparameter optimization via Bayesian sweep
5. Deploy optimized model on RTX 4060 via TensorRT
6. Create real-world validation dataset (fsoco-ubm)

1.4 Contributions

We evaluated three YOLO architectures and found that YOLO26n, the most recent and lightest, reaches mAP$_{50}$=0.763 on the FSOCO-12 benchmark—a 14.6% improvement over the model currently running on the car—while achieving 2.63 ms inference on the onboard RTX 4060. We also created fsoco-ubm, a real-world test set from the car's own camera, which revealed a 22–27% accuracy gap between internet benchmarks and deployment conditions across all models tested.

Chapter 2

Background

2.1 Object Detection with YOLO

YOLO (You Only Look Once) is a family of real-time object detection models first introduced by Redmon et al. (2016) and now maintained by Ultralytics (Jocher et al., 2023). Unlike two-stage detectors that first propose regions and then classify them, YOLO performs detection in a single forward pass, which is what makes it fast enough for our use case. We evaluated three successive generations of nano-sized YOLO models.

2.1.1 YOLOv11

YOLOv11 was released in 2024 by Ultralytics (Ultralytics, 2024) and is the architecture currently deployed on the UBM race car. It introduces C3k2 backbone blocks, a Spatial Pyramid Pooling Fast (SPPF) layer, and C2PSA attention modules. The nano variant has 2.59M parameters and 6.4 GFLOPs. This is our baseline: every other model in this report is measured against it.

2.1.2 YOLO12

YOLO12, published in January 2025 by Tian et al. (2025), replaces the CSP backbone with R-ELAN (Residual Efficient Layer Aggregation Network) and introduces an area-attention mechanism that captures global context without the quadratic cost of standard self-attention. The nano variant sits at 2.56M parameters and 6.3 GFLOPs, marginally smaller than YOLOv11n.

2.1.3 YOLO26

YOLO26 is the most recent release from Ultralytics at the time of writing (Ultralytics, 2025). Its nano variant is the smallest of the three we evaluated: 2.51M parameters and

5.8 GFLOPs. It also eliminates the non-maximum suppression (NMS) post-processing step, further reducing latency.

2.2 Transfer Learning and Catastrophic Forgetting

Transfer learning is the practice of pre-training a model on a large dataset and then fine-tuning it on a smaller, task-specific one (FSB Driverless, 2024; FMDV, 2024). The main risk is catastrophic forgetting (French, 1999): if the learning rate is too high during fine-tuning, the model overwrites the useful features it learned during pre-training. Standard mitigation strategies include freezing early layers, using a low learning rate, and warming up gradually. We encountered this problem during our two-stage training and document the diagnosis and fix in sections A.3 and 3.6.

2.3 Model Optimization for Edge Deployment

The models are trained in PyTorch but deployed through a multi-stage optimization pipeline. First, the trained weights are exported to ONNX (Open Neural Network Exchange) (ONNX Community, 2019), a hardware-agnostic intermediate representation. Then, NVIDIA TensorRT (NVIDIA Corporation, 2024) compiles the ONNX graph into an optimized engine for the target GPU. TensorRT applies several automatic optimizations: kernel fusion combines sequences of operations (e.g. Conv+BatchNorm+ReLU) into a single GPU kernel to reduce memory bandwidth and kernel launch overhead, and precision reduction converts FP32 weights to FP16 or INT8 to exploit the GPU's dedicated Tensor Cores. INT8 quantization requires a calibration dataset of 500–1,000 representative images to determine the optimal scaling factors, and typically incurs a 2–3% mAP_{50} drop compared to FP16.

One important constraint is that TensorRT engines are tied to the specific GPU architecture they are built on. An engine compiled for the RTX 4080 Super (training machine) will not run on the RTX 4060 (race car), so the final engine must be built on the target hardware.

2.4 Formula Student Driverless Perception

The perception pipeline, as designed by Fusa (2025), is built around a ZED 2i stereo camera (Stereolabs, 2021) operating at 1280×720 resolution and up to 60 fps. The YOLO detector runs on each frame to produce 2D bounding boxes for five cone classes defined by the FSAE Driverless rules (SAE International, 2025): small blue cones mark the left track border, small yellow cones mark the right border, small orange cones delineate entry and exit lanes, and large orange cones are placed before and after start, finish, and timekeeping lines. A fifth class, “unknown,” is used for ambiguous or occluded cones that cannot be confidently classified. These detections are then matched across the stereo pair and triangulated to produce 3D positions relative to the car. The entire pipeline must complete within 16.7 ms per frame to maintain the 60 fps loop rate; any excess time taken by detection directly reduces the budget available for planning and control downstream.

2.5 Related Work

The FSOCO (Formula Student Objects in Context) dataset (FSOCO Contributors, 2020) is a community-maintained benchmark for cone detection, contributed by multiple Formula Student teams. It provides a shared evaluation standard, though the images come from a variety of cameras and conditions that may not match any single team’s deployment setup. Most teams in the Driverless competition use some variant of YOLO, trained on FSOCO or private datasets, and evaluate using the standard COCO metrics (Lin et al., 2014): mAP_{50} (intersection-over-union threshold of 0.5) and $\text{mAP}_{50\text{-}95}$ (averaged over thresholds from 0.5 to 0.95). The prior work at UBM is documented in the thesis by Fusa (2025), which established the YOLOv11n baseline and the stereo matching pipeline we build upon.

Chapter 3

Methods

3.1 Experimental Setup

We used a M1 MacBook Air for code development and an Ubuntu workstation with an RTX 4080 Super for training. The existing inference pipeline (`ubm-yolo-detector`) enforced the use of Ultralytics models, which we kept to slot the new model into the existing modular framework. Weights and Biases (Weights & Biases, Inc., 2024) provided experiment tracking. The Autonomous System Unit (ASU) on the race car is a water-cooled desktop running ROS2 (Fusa, 2025); for our purposes we care only about its GPU, an RTX 4060.

Table 3.1: Hardware used for training and deployment.

Machine	**Purpose**	**GPU**	**OS**
M1 MacBook	Code development	CPU only	macOS
Ubuntu Workstation	Training & testing	RTX 4080 Super (CUDA)	Ubuntu
ASU (Race Car)	Deployment	RTX 4060	Ubuntu (ROS2)

3.2 Datasets

We used three datasets, summarized in table 3.2. FSOCO-12 (FMDV, 2024) is the primary training and benchmarking dataset. The cone-detector dataset (FSB Driverless, 2024) is a larger collection used exclusively for pre-training in the two-stage strategy. Finally, fsoco-ubm is a small test set we created from the car's own camera to validate whether benchmark results translate to the real world.

Table 3.2: Datasets used in this project.

Dataset	Images	Instances	Split	Purpose
FSOCO-12 (train)	7,120	∼78,000	Train	Primary training
FSOCO-12 (val)	1,968	∼36,000	Validation	Model selection
FSOCO-12 (test)	689	12,054	Test	Standard benchmark
cone-detector	22,725	∼200,000	Train	Stage 1 pre-training
fsoco-ubm	96	1,426	Test	Real-world validation

3.2.1 FSOCO-12

FSOCO-12 (Formula Student Objects in Context, version 12) is a community-curated dataset hosted on Roboflow (FMDV, 2024; FSOCO Contributors, 2020). It contains 9,777 images split into train (7,120), validation (1,968), and test (689) sets, with approximately 126,000 cone instances across the five FSAE classes. The images come from multiple Formula Student teams and cover diverse lighting, weather, and track conditions. The class distribution is imbalanced: yellow and blue cones dominate (∼77% of test instances), while unknown cones account for only 5.6% and are the hardest to detect. We use the test set (689 images, 12,054 instances) as our primary benchmark throughout this report.

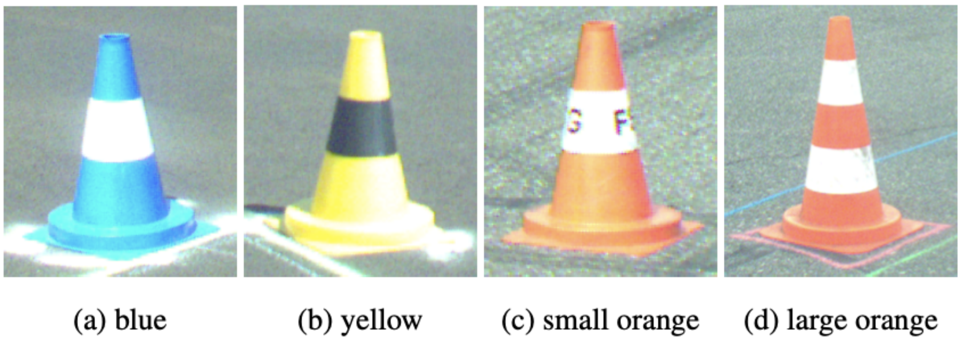

Figure 2: FSOCO supports five object classes. The four main classes are shown here, the fifth class *other* includes all cones that are not rules compliant.

Figure 3.1: Class distribution in the FSOCO-12 dataset.

3.2.2 cone-detector

The cone-detector dataset (FSB Driverless, 2024) contains 22,725 images with the same five cone classes as FSOCO-12, making it 2.3× larger. It is also a community dataset from Roboflow, contributed by a different Formula Student team. Training a model from scratch on this dataset alone plateaus at $mAP_{50} \approx 0.68$, which is worse than our FSOCO-12 baseline. However, the volume of data makes it useful for pre-training before fine-tuning on the curated FSOCO-12 set (section 3.6).

3.2.3 fsoco-ubm: Real-World Test Set

FSOCO-12 is an internet dataset assembled from many teams and cameras, so good performance on it does not guarantee good performance on our car. To address this, we created fsoco-ubm: a 96-image test set extracted directly from the ZED 2i stereo camera on the UBM race car. The images were recorded on November 20, 2025 at the Rioveggio test track during driving runs at 30–50 km/h. We extracted one frame every 60 frames from the stereo video (one sample every two seconds of real-world time), split the 2560×720 stereo pairs into left and right images of 1280×720, and annotated them using Roboflow Label Assist with manual review.

The dataset contains 1,426 cone instances across the five classes, dominated by yellow and blue cones with very few orange instances. The real-world conditions introduce challenges absent from the internet benchmark: motion blur, variable outdoor lighting, small pixel area for distant cones, and color desaturation. We use fsoco-ubm strictly as a held-out test set and never for training.

3.3 Baseline Reproduction

Our first step was to reproduce the baseline from Fusa's thesis (Fusa, 2025), which reports mAP_{50}=0.824 for YOLOv11n on FSOCO. We trained YOLOv11n for 300 epochs on FSOCO-12 using Ultralytics default hyperparameters (AdamW optimizer, lr0=0.01, batch 64, 640×640 input, default augmentation with mosaic=1.0, mixup=0.0). The model converged around epoch 200 and plateaued at mAP_{50}=0.714 on the validation set, 13.4% below the thesis claim.

We contacted the thesis author, who confirmed that the reported results were produced by other team members (Gabriele and Patta) using an unknown "particular dataset" and training configuration that was subsequently lost. The production model currently deployed on the car, which we evaluated independently on the FSOCO-12 test set, achieves only mAP_{50}=0.666. Our baseline therefore already exceeds the production model by 6.2%, and we adopt mAP_{50}=0.707 (test set) as the reproducible reference point for all subsequent experiments.

3.4 Hyperparameter Sweep

To determine whether the baseline could be improved through hyperparameter tuning, we ran a Bayesian optimization sweep using Weights & Biases (Weights & Biases, Inc., 2024; Snoek et al., 2012). The sweep explored 13 hyperparameters: learning rate (lr0, lrf), momentum, weight decay, warmup epochs, close mosaic epoch, dropout, and six augmentation parameters (hsv_h, hsv_s, hsv_v, mosaic, mixup, copy_paste). Each run trained for 100 epochs on FSOCO-12 with YOLOv11n.

Of 21 planned runs, 10 completed successfully and 11 crashed. The crashes were strongly correlated with aggressive augmentation: crashed runs had 4× higher mixup (0.197 vs 0.049) and nearly 2× higher dropout (0.156 vs 0.081) on average. High mixup creates heavily blended training images that, combined with high dropout, introduce too much regularization for the model to converge. Among the 10 completed runs, the best achieved mAP_{50}=0.709, 0.7% *worse* than our baseline (table 3.3). The mean was 0.703 with a standard deviation of 0.019, confirming that YOLOv11n on FSOCO-12 is insensitive to these hyperparameters. We stopped the sweep and pivoted to architecture changes.

Table 3.3: Hyperparameter sweep summary (W&B Bayesian optimization).

Metric	**Value**
Runs completed	10 / 21
Best sweep mAP_{50}	0.709
Baseline mAP_{50}	0.714
Mean of sweep runs	0.703
Standard deviation	0.019

3.5 Architecture Evaluation

Since hyperparameter tuning proved ineffective, we evaluated whether newer architectures could break through the YOLOv11n performance ceiling. We trained YOLO12n (Tian et al., 2025) and YOLO26n (Ultralytics, 2025) under identical conditions: FSOCO-12 dataset, 300 epochs, batch 64, Ultralytics default hyperparameters. All three models are nano variants with comparable parameter counts (2.51–2.59M) and GFLOPs (5.8–6.4), as shown in table 3.4, making the comparison fair with respect to model capacity. YOLO12n finished at mAP_{50}=0.708 on the test set, a marginal gain over YOLOv11n (0.707). YOLO26n reached mAP_{50}=0.763, a substantial 7.9% improvement over our baseline and 14.6% over the UBM production model. Every class improved, with the largest absolute gains on unknown cones (+23.4%) and orange cones (+6.8%).

Table 3.4: Architecture specifications for the three evaluated models.

Model	**Params (M)**	**GFLOPs**	**Year**
YOLOv11n	2.59	6.4	2024
YOLO12n	2.56	6.3	2025
YOLO26n	2.51	5.8	2025

3.6 Two-Stage Training Strategy

Given that YOLO26n achieved the best single-stage result, we investigated whether pre-training on a larger dataset could push it further. The cone-detector dataset (FSB Driverless, 2024) contains 22,725 images with the same five classes as FSOCO-12, providing 2.3× more training data. Both datasets target the same task—cone detection—so the domain gap is small.

Stage 1 pre-trained YOLO26n on cone-detector starting from COCO-pretrained weights. Training was planned for 400 epochs but the loss converged early and we stopped at epoch 338, achieving mAP_{50}=0.734 on the cone-detector validation set. Stage 2 fine-tuned the Stage 1 checkpoint on FSOCO-12 for 300 additional epochs. Our first attempt at Stage 2 failed: the model's mAP_{50} dropped from 0.754 to 0.628 in just two epochs—catastrophic forgetting (French, 1999). The root cause was Ultralytics' `optimizer='auto'` default, which silently replaced our specified learning rate of 0.001 with a hardcoded 0.01, as detailed in section A.3.

We redesigned Stage 2 as a two-phase process using explicit AdamW (Loshchilov and Hutter, 2019). Phase 2A froze the first 10 backbone layers and trained only the detection head for 50 epochs with lr=0.001 and 20% warmup (10 epochs). Phase 2B then unfroze all layers for 250 epochs with an ultra-low learning rate of 0.00005 (100× lower than training from scratch) and 50 epochs of warmup with cosine annealing. This eliminated the forgetting entirely: validation mAP_{50} improved monotonically throughout Stage 2 and converged at 0.761 on the FSOCO-12 test set, comparable to the single-stage result (0.763) but with better recall and real-world precision, as we discuss in chapter 4.

3.7 Deployment Pipeline

The trained PyTorch model is deployed through a three-step export pipeline. First, the `best.pt` checkpoint is exported to ONNX (ONNX Community, 2019) with a fixed batch size of 2 (two images in a stereo pair). Then, on the ASU itself, the ONNX graph is compiled into a TensorRT FP16 engine (NVIDIA Corporation, 2024) using `trtexec`. Building the engine on the target hardware is mandatory because TensorRT optimizes kernel selection and memory layout for the specific GPU architecture (RTX 4060).

The C++ ROS2 inference node (`ros_yolo_detector_node`) receives stereo image pairs from the ZED 2i camera topic and runs YOLO inference on the left and right frames in parallel using two threads. The resulting bounding box vectors are sorted by center coordinates and matched across the stereo pair using a distance and similarity criterion, followed by feature matching for robust association. Matched detections are triangulated into 3D positions in the camera coordinate frame and then transformed into the vehicle frame (FLU convention) for downstream SLAM consumption. The node publishes both an annotated image with bounding box overlays and a `DetectionInfosArray` message containing 3D coordinates and confidence scores. We lowered the detection confidence threshold from the default 0.5 to 0.35, which improved mAP$_{50}$ on fsoco-ubm by 10% by recovering additional true positives at the cost of a marginal increase in false detections. This parameter as well as the NMS threshold are modifiable without recompiling and can be tuned further on the track.

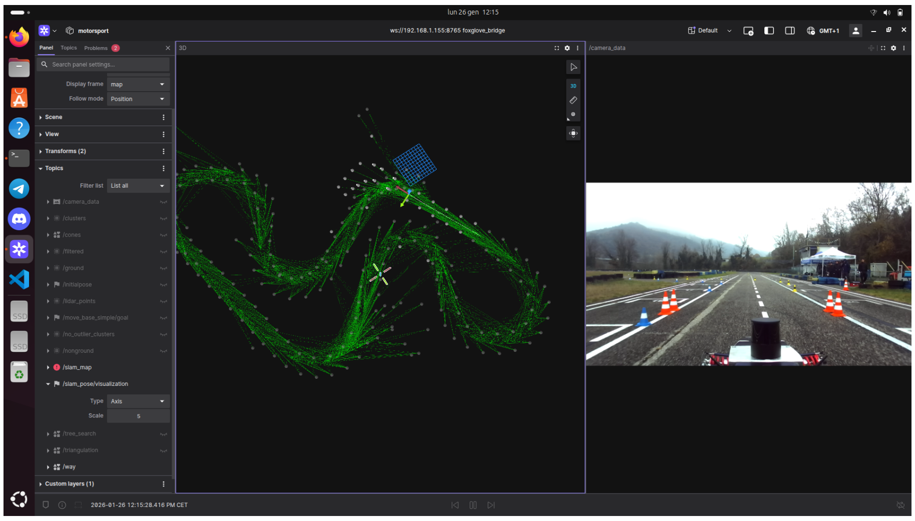

Figure 3.2: 3D visualization of detected cones from the stereo pipeline.

Chapter 4

Results

We evaluate all models on two test sets: the FSOCO-12 benchmark (689 images) and fsoco-ubm (96 images from the car's own camera), reporting standard COCO metrics (Lin et al., 2014).

4.1 FSOCO-12 Benchmark Results

Table 4.1 presents the FSOCO-12 test set results. YOLO26n dominates: the single-stage variant achieves the highest mAP_{50}=0.763 and precision (0.849), while the two-stage variant leads in recall (0.708) and $\text{mAP}_{50\text{-}95}$=0.528. Both YOLO26n variants outperform the UBM production model by over 14%, confirming that the architecture upgrade accounts for the bulk of the improvement. YOLO12n and our retrained YOLOv11n baseline land nearly identically at $\text{mAP}_{50}\approx 0.708$, both already 6% above the production model.

Table 4.1: Overall performance on the FSOCO-12 test set (689 images, 12,054 instances).

Model	**Training Data**	**Epochs**	**mAP_{50}**	**$\text{mAP}_{50\text{-}95}$**	**Prec.**	**Recall**
YOLO26n (single)	FSOCO-12	300	**0.763**	0.524	**0.849**	0.694
YOLO26n (two-stage)	CD→FSOCO-12	338+300	0.761	**0.528**	0.832	**0.708**
YOLO12n	FSOCO-12	300	0.708	0.485	0.840	0.654
YOLOv11n (ours)	FSOCO-12	300	0.707	0.490	0.816	0.662
UBM Production	unknown	300	0.666	0.461	0.803	0.579

CD = cone-detector dataset (22,725 images, pre-training stage).

4.2 Per-Class Analysis

Table 4.2 shows a consistent per-class hierarchy. Large orange cones are easiest (mAP_{50}=0.886), followed by blue (0.863), yellow (0.856), and orange (0.843). Unknown cones remain the hardest class at mAP_{50}=0.364—these are cones that annotators could not confidently classify, so the model inherits that ambiguity. YOLO26n improves every class over YOLO12n, with the largest gain on unknown cones (+23.4%).

Table 4.2: Per-class performance of YOLO26n (single-stage) on FSOCO-12 test set.

Class	**Images**	**Instances**	**Prec.**	**Recall**	**mAP$_{50}$**	**mAP$_{50\text{-}95}$**
Large Orange Cone	154	408	0.873	0.833	**0.886**	0.688
Blue Cone	506	4,437	0.927	0.783	0.863	0.602
Yellow Cone	562	4,844	0.915	0.774	0.856	0.583
Orange Cone	286	1,686	0.892	0.779	0.843	0.571
Unknown Cone	68	679	0.635	0.297	0.364	0.178

4.3 Two-Stage vs. Single-Stage Training

Table 4.3 compares the two training strategies head-to-head. On the FSOCO-12 benchmark the difference is within noise: mAP_{50} differs by just 0.2%, and neither variant is consistently better across all metrics. The two-stage model wins on recall (+1.4 percentage points) and $mAP_{50\text{-}95}$ (+0.6%), at the cost of lower precision (−1.6 percentage points). On the real-world fsoco-ubm set, the two models tie at mAP_{50}=0.565, but two-stage achieves 3.4 percentage points higher precision, indicating fewer false positives under deployment conditions. We deployed the two-stage model because missing a cone is more dangerous than a false detection in autonomous racing. However, we do not recommend training two stage models due to their quadrupled computational cost; better to train other architectures and improve the datasets.

Table 4.3: Two-stage vs. single-stage YOLO26n training comparison.

Metric	**Single-Stage**	**Two-Stage**	**Delta**
FSOCO-12 mAP$_{50}$	**0.763**	0.761	−0.2%
FSOCO-12 mAP$_{50\text{-}95}$	0.524	**0.528**	+0.6%
FSOCO-12 Precision	**0.849**	0.832	−1.6 pp
FSOCO-12 Recall	0.694	**0.708**	+1.4 pp
fsoco-ubm mAP$_{50}$	0.565	**0.565**	+0.04%
fsoco-ubm Precision	0.615	**0.649**	+3.4 pp
Generalization Gap	−25.9%	**−25.8%**	+0.1 pp

pp = percentage points.

4.4 Real-World Validation (fsoco-ubm)

Table 4.4 shows performance on fsoco-ubm. Every model suffers a substantial drop from the FSOCO-12 benchmark, ranging from −21.5% (YOLOv11n) to −27.0% (YOLO12n). This gap reflects the harder real-world conditions described in section 3.2.3. Despite the drop, the YOLO26n variants maintain first place at mAP$_{50}$=0.565. An interesting finding is that YOLOv11n generalizes best, losing only 21.5% and achieving the highest precision on this set (0.874), which suggests that its simpler architecture is more conservative and less prone to false positives on out-of-distribution data. YOLO12n generalizes worst among our trained models (−27.0%) and ends up tied with UBM production at 0.517.

Table 4.4: Real-world performance on fsoco-ubm test set (96 images, 1,426 instances).

Model	**mAP$_{50}$**	**Prec.**	**Recall**	**Gap vs FSOCO-12**
YOLO26n (two-stage)	**0.565**	**0.649**	0.462	−25.8%
YOLO26n (single)	**0.565**	0.615	**0.469**	−25.9%
YOLOv11n (ours)	0.555	0.874	0.447	−21.5%
YOLO12n	0.517	0.572	0.454	−27.0%
UBM Production	0.517	0.635	0.393	−22.3%

4.5 Deployment Performance

Table 4.5 breaks down inference on the RTX 4060 onboard the race car. YOLO26n averages 2.63 ms per image—transfer-bound, with only 1.02 ms of GPU compute—leaving a 6.3× margin over the 60 fps budget. We did not pursue INT8 quantization: the 1–2% mAP$_{50}$ penalty is not justified when the model is already well within the real-time envelope.

Table 4.5: Inference latency breakdown on RTX 4060 (TensorRT FP16).

Component	**YOLO26n (ms)**	**YOLOv11n prod. (ms)**
H2D Transfer	1.58	1.58
GPU Compute	**1.02**	0.99
D2H Transfer	0.03	0.13
Total	**2.63**	2.70
Max FPS	380	370

Chapter 5

Conclusion

5.1 Summary of Contributions

Of the three architectures evaluated, YOLO26n achieved the best results: mAP_{50}=0.763 on FSOCO-12 (+14.6% over the previous production model) and 2.63 ms inference on the onboard RTX 4060, leaving a 6.3× margin for 60 fps operation. A two-stage training variant traded marginal benchmark precision for better recall and real-world generalization. Hyperparameter tuning proved ineffective. Our real-world test set, fsoco-ubm, revealed a 22–27% accuracy drop from the internet benchmark across all models—a gap that would have gone unnoticed without car-specific validation data.

5.2 Lessons Learned

Contributing to the existing codebase was straightforward thanks to the modular software architecture established by Fusa (2025): detection, stereo matching, and triangulation are separate stages, so swapping the YOLO model required only replacing the TensorRT engine and updating the confidence threshold. This report also fills a documentation gap: the previous training configuration was lost when team members graduated, so we have documented every run, dataset version, and export step for reproducibility. The key technical lessons are:

- **Architecture selection matters more than hyperparameter tuning.** The sweep explored 13 parameters across 21 runs and found no improvement over defaults, while upgrading from YOLOv11n to YOLO26n gained 7.9% mAP_{50}.
- **Always validate on real data.** FSOCO-12 performance does not reliably predict deployment performance: all models lost 22–27% on our car's camera data. Without fsoco-ubm, we would have overestimated the deployed model's accuracy.

- **Two-stage training provides marginal gains at this data scale.** Pre-training on 3× more data improved recall and real-world precision, but the benchmark mAP_{50} remained within 0.2% of single-stage for 4× the compute.
- **Verify framework internals.** The Ultralytics `optimizer='auto'` setting silently overwrites user-specified learning rates, which caused catastrophic forgetting during fine-tuning and cost a full day of debugging (section A.3).

5.3 Limitations

Several limitations should be noted. We could not reproduce the mAP_{50}=0.824 reported in Fusa (2025): the training configuration and dataset version used to produce that figure were lost, and the best we achieved with YOLOv11n on FSOCO-12 under default hyperparameters was 0.707, which we adopted as our reproducible baseline. The fsoco-ubm test set is small (96 images) and captures a single track on a single day, so it does not cover the full range of conditions the car will face at competition—different track layouts, rain, and dusk lighting are absent. The unknown cone class remains poorly detected (mAP_{50}=0.364), a dataset-level problem rooted in annotator ambiguity. We only evaluated nano-sized models (∼2.5M parameters); larger variants may improve accuracy within the available latency budget. Finally, we did not ablate individual augmentation parameters.

5.4 Future Work

We recommend the following directions for continued development:

- **Evaluate larger YOLO26 variants.** The 6.3× latency margin leaves room for the small or medium variants, which may improve detection of distant and ambiguous cones. Even a model 3× slower would remain real-time.
- **Expand fsoco-ubm.** Add images from different tracks, weather conditions, and times of day. The current 96 images from a single session are a starting point, not a representative benchmark.
- **Target the unknown cone class.** Data augmentation strategies such as color jittering and occlusion simulation may help the model learn to flag uncertain detections. Alternatively, merging unknown cones into the nearest color class during training could improve overall recall at the cost of classification granularity.
- **Auto-exposure compensation.** The ZED 2i camera's auto-exposure underexposes cones when the sky dominates the frame. A preprocessing step that adjusts exposure based on the lower half of the image could reduce this failure mode.

- **End-to-end integration testing.** We validated detection accuracy and inference speed independently but did not test the full autonomous stack (detection $\rightarrow$ matching $\rightarrow$ triangulation $\rightarrow$ SLAM $\rightarrow$ planning $\rightarrow$ control) with the new model under race conditions.

UBM will compete at Formula Student Germany from 11–16 August 2026. The goal is to complete all four dynamic events—Trackdrive, Autocross, Skid Pad, and Acceleration—for the first time; last year a broken axle ended the campaign before we could attempt all of them. The priority is reliability over outright speed: finishing every event and collecting data is more valuable than optimizing for a single fast lap, because the data enables iteration. The improvements in this report contribute to that goal by providing a more accurate and faster perception pipeline, but the real test will come at competition.

Appendix A

Technical Details

A.1 Hyperparameter Sweep Configuration

```
# TODO: Paste actual sweep YAML from wandb_sweep.yaml
method: bayes
metric:
  name: metrics/mAP50(B)
  goal: maximize
parameters:
  lr0:
    min: 0.001
    max: 0.05
  momentum:
    min: 0.8
    max: 0.98
  # ... (add remaining 11 parameters)
```

Listing A.1: W&B Bayesian sweep configuration.

A.2 Two-Stage Training Hyperparameters

A.3 The `optimizer='auto'` Bug

A.4 fsoco-ubm Dataset Creation Pipeline

```
# TODO: Paste key extract from extract_frames_from_avi.py
import cv2

cap = cv2.VideoCapture("media/lidar1.avi")
```

Table A.1: Two-stage training configuration.

Parameter	**Stage 1 (cone-detector)**	**Stage 2 (FSOCO-12)**
Dataset	cone-detector (22,725 imgs)	FSOCO-12 (5,536 imgs)
Epochs	338 (converged early)	300
Batch size	64	64
Optimizer	auto (SGD)	AdamW
Learning rate	0.01	0.001
Image size	640	640
Freeze layers	None	Phase-dependent
Pretrained weights	COCO (`yolo26n.pt`)	Stage 1 `best.pt`

```python
frame_count = 0
saved = 0
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    if frame_count % 60 == 0:  # every 2 seconds
        cv2.imwrite(f"ubm_test_set/images/lidar1_{saved:04d}.png"
           , frame)
        saved += 1
    frame_count += 1
cap.release()
```

Listing A.2: Frame extraction from AVI video.

Bibliography

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. doi: 10.1007/s11263-009-0275-4.

FMDV. FSOCO cone detection dataset (version 12), 2024. URL `https://universe.roboflow.com/fmdv/fsoco-kxq3s/dataset/12`. Roboflow Universe. Accessed January 2026.

Formula Student Germany. Formula Student Germany — driverless, 2024. URL `https://www.formulastudent.de/fsg/`. Accessed January 2026.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. doi: 10.1016/S1364-6613(99)01294-2.

FSB Driverless. Cone detector dataset (version 1), 2024. URL `https://universe.roboflow.com/fsbdriverless/cone-detector-zruok/dataset/1`. 22,725 images. Roboflow Universe. Accessed January 2026.

FSOCO Contributors. FSOCO: Formula student objects in context, 2020. URL `https://www.fsoco-dataset.com/`. Community dataset for Formula Student cone detection.

Edoardo Fusa. Pushing cars' limits: Exploring autonomous technologies in the formula SAE driverless competition. Master's thesis, University of Bologna, 2025. Stereocamera pipeline for autonomous racing.

Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLOv8, 2023. URL `https://github.com/ultralytics/ultralytics`. Accessed January 2026.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. *arXiv preprint arXiv:1405.0312*, 2014.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

NVIDIA Corporation. NVIDIA TensorRT: Programmable inference accelerator, 2024. URL https://developer.nvidia.com/tensorrt. Accessed January 2026.

ONNX Community. ONNX: Open neural network exchange, 2019. URL https://onnx.ai/. Accessed January 2026.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

Roboflow, Inc. Roboflow: Computer vision tools, 2024. URL https://roboflow.com/. Dataset management and annotation. Accessed January 2026.

SAE International. FSAE driverless supplement v11, 2025. URL https://www.fsaeonline.com/CompResources/2025/8f030a58-d9e4-49b8-bc83-6ca16c7ce715/FSAE-Driverless-Supplement-V11.pdf. Section: Dynamic Event Markings.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, 2012.

Stereolabs. ZED 2i stereo camera, 2021. URL https://www.stereolabs.com/products/zed-2i. Accessed January 2026.

Yunjie Tian, Qiang Ye, and David Doermann. YOLO12: Attention-centric real-time object detectors. *arXiv preprint arXiv:2502.12524*, 2025.

Ultralytics. YOLO11: Real-time object detection, 2024. URL https://docs.ultralytics.com/models/yolo11/. Accessed January 2026.

Ultralytics. YOLOv26: Advancing real-time object detection, 2025. URL https://docs.ultralytics.com/models/yolo26/. Accessed January 2026.

Weights & Biases, Inc. Weights & Biases, 2024. URL https://wandb.ai/. Experiment tracking platform. Accessed January 2026.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2018.