Dynamic Resolution Adaptation for the Vision Pipeline in Autonomous Driving

Mustafa Göktan Güdükbay
The Pennsylvania State University
State College, PA, USA
mfg5472@psu.edu

Wanhang Lu
The Pennsylvania State University
State College, PA, USA
wvl5336@psu.edu

Kiwan Maeng
The Pennsylvania State University
State College, PA, USA
kvm6242@psu.edu

Mahmut Taylan Kandemir
The Pennsylvania State University
State College, PA, USA
mtk2@psu.edu

Anand Sivasubramaniam
The Pennsylvania State University
State College, PA, USA
axs53@psu.edu

Abstract

Multi-GPU setups can help meet real-time latency and accuracy requirements for autonomous driving, but introduce power, cooling, and storage challenges that reduce driving range. Ideally, a single processor is preferable for budget and energy efficiency, but it raises latency concerns. We focus on multi-camera 2D object detection using the YOLOv9 model on a single-processor setup, aiming to minimize overall latency while maintaining detection accuracy. We propose a dynamic scheduling framework that predicts the required resolution for the next frame, given the necessary accuracy threshold, using an ordinal regression model. We compare static scheduling, an oracle baseline that assumes prior knowledge of the accuracy, and dynamic scheduling based on predictions from previous frames. On the Jetson platform, our scheduler achieves lower latency while maintaining comparable accuracy to fixed-resolution baselines.

CCS Concepts

• **Computer systems organization → Neural networks**.

Keywords

autonomous driving, object detection, You Only Look Once (YOLO), scheduling, latency, power consumption, Graphics Processing Unit (GPU).

Reference Format:
Mustafa Göktan Güdükbay, Wanhang Lu, Kiwan Maeng, Mahmut Taylan Kandemir, and Anand Sivasubramaniam. 2026. Dynamic Resolution Adaptation for the Vision Pipeline in Autonomous Driving. In *Proceedings of Workshop on Systems and Architectures for Neural Rendering, AR/VR, and Visual Computing (VisArch '26)*, 5 pages.

1 Introduction

Autonomous driving typically relies on deep learning (DL) for perception tasks such as object detection. Each sensor produces multiple inputs per second, and detection networks must satisfy real-time latency requirements because the control and path-planning modules depend on these inference results. Most solutions are deployed across multiple high-end accelerators, such as the NVIDIA Drive series. Although the exact hardware used in commercial deployments is not disclosed, it is reasonable to assume that autonomous vehicles require substantial resources, likely multiple GPUs, to achieve real-time performance for high-accuracy perception.

While multi-GPU setups can fulfill latency and accuracy requirements for high-resolution models, each additional GPU results in approximately a 3-4% reduction in driving range due to overheads such as cooling and memory [4]. In single-processor setups, we encounter latency issues with high-resolution models or accuracy issues with low-resolution models. Thus, we explore various scheduling strategies for real-time 2D object detection to balance accuracy and latency without resorting to multi-processor setups. In our experiments, we use the YOLOv9 object detection framework, and the dynamic scheduler we propose predicts the lowest input resolution needed to meet a target accuracy threshold for each frame, improving both latency and accuracy [14]. We compare our dynamic scheduler against two baseline schedulers: a static scheduler that uses the same resolution for all frames, and an oracle scheduler that selects the smallest resolution that satisfies the target accuracy for each frame.

We evaluate scheduling schemes to analyze trade-offs between accuracy and latency, using videos from the Waymo Perception dataset [11], including 20-sec. videos from five cameras. For easier videos, our dynamic scheduler reduces latency on Jetson Nano by 47.6% compared to the fixed 1024×1024 resolution and by 66.4% compared to the fixed 1280×1280 resolution. For medium-difficulty videos, latency drops by 24.5% compared to the fixed 1024×1024 resolution and by 51.6% compared to the fixed 1280×1280 resolution. In challenging videos, our dynamic scheduler matches the accuracy of 1024×1024 while achieving lower latency in some cases, prioritizing specific cameras as needed. The contributions are as follows:

- We observe variability in detection accuracy across camera frames and the impact of frame resolution.
- We introduce an ordinal regression model to predict the resolution for the next frame based on a set of statistical features on detected objects from the previous frame.
- We compare the performance of proposed scheduling algorithms using the Waymo Perception dataset [11].
- Our prediction-based dynamic scheduler achieves near-optimal trade-offs among latency, accuracy, and energy consumption across a diverse set of videos with varying object-detection difficulty, comparable to those of an oracle system.

2 Background and Related Work

Our work targets the perception pipeline in autonomous driving using a modular approach similar to Autoware [1] and Waymo [16]. This approach organizes computations into sequential modules: *sensory input processing*, *perception*, *localization*, *control*, and *command*, with each module building on the previous one [3]. We integrate computer vision and neural networks for 2D object detection. A typical Autoware configuration includes seven cameras and three

LiDAR inputs. Six cameras are used for 2D object detection, providing a full 360-degree field of view, while the seventh camera is dedicated to traffic light detection using a YOLO-based network. The LiDAR point cloud data is fused to detect occluded objects that may be missed by 2D detectors. Six identical YOLO networks run in parallel for 2D object detection, and a CenterPoint neural network [18] processes 3D point cloud data on a separate stream.

Several factors influence the design and manufacturing of autonomous vehicles, including computational resources, energy consumption, peak power demands, inference models for accuracy, and latency to meet real-time constraints [9]. Lin et al. [4] examine the energy, power, and latency trade-offs of the autonomous driving software stack using various hardware configurations. As computing costs rise, cooling costs increase, thereby increasing overall power consumption.

Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), custom Deep Learning Accelerators (DLAs), and Central Processing Units (CPUs) are used for DL inference, each with advantages and drawbacks in latency, cost, power, and energy use. GPUs, designed for Single-Instruction, Multiple-Data (SIMD) instructions, are the most common accelerators for DL. High-end and embedded GPUs are frequently utilized for this purpose. NVIDIA has proposed specialized systems, such as the NVIDIA DRIVE series [8], that aim to be more power-efficient than traditional server architectures while maintaining equivalent performance. Other Edge DL hardware architectures for autonomous driving include Nvidia RTX [7], Jetson AGX Orin [6], Coral Edge TPU [2], and Xilinx ZCU102 FPGA [17].

3 Motivation

We study i) latency variations of different resolutions on three hardware architectures, ii) accuracy changes for each camera over time with different models using different resolutions, and iii) the effect of scheduling algorithms on impacting the desired metrics of concern (accuracy, energy, and latency). We use the observed tradeoffs to motivate the need for a dynamic scheduling mechanism. We use per-video averages to evaluate accuracy, latency, error, and pass rate.

Model – Image Resolution and Precision. Table 1 shows the specifications for the Front (F), Front-Left (FL), Front-Right (FR), Side-Left (SL), and Side-Right (SR) cameras in the Waymo Perception Dataset [11]. Image sizes are determined by cropping and downsampling the original data, while the horizontal field of view (HFOV) denotes the angular range. The dataset comprises 20-second video sequences recorded at 10 Hz, synchronized across five camera angles. Videos are categorized into three difficulty levels based on the Static-1280 scheduler's pass rate: hard (0-50%), medium (50-75%), and easy (75-100%), with counts of 60 easy, 90 medium, and 44 hard videos.

Table 1: Camera specifications.

Camera	F	FL, FR	SL, SR
Image Size	1920 × 1280	1920 × 1280	1920 × 1040
HFOV	±25.2°	±25.2°	±25.2°

We utilize three metrics to evaluate the object detection accuracy [13]: mean Average Precision (*mAP*) at an Intersection over Union (IoU) threshold of 0.5 (*mAP*@0.5), *mAP* at IoU thresholds from 0.5 to 0.95 with 0.05 increments (*mAP*@0.5−0.95), and *mAP* from the Waymo Open Dataset Challenge, where IoU threshold is 0.7 for vehicles and 0.5 for pedestrians and cyclists (*mAP*@) [19].

Table 2 shows the performance and power statistics for Yolov9 with different resolutions on three NVIDIA architectures: RTX3060, RTX3060 Ti, and Jetson AGX Orin. All models use floating-point 16 (FP16) precision and were exported to TensorRT for fast inference. Across all resolutions, the Jetson AGX Orin is 2-3× slower than other GPUs because it is optimized for power efficiency; the RTX 3060 and RTX 3060 Ti exhibit similar latencies at the same resolution.

Table 2: Performance and power statistics. The resolutions are square; i.e., 256 means 256×256.

Resolution	Hardware	*mAP*@ 0.5	*mAP*@ [0.5−0.95]	*mAP*	Latency (ms)
256	RTX3060	0.220	0.128	0.202	1.289
256	Jetson AGX Orin	0.220	0.128	0.202	2.620
256	RTX3060 Ti	0.220	0.128	0.202	1.344
512	RTX3060	0.346	0.211	0.324	2.072
512	Jetson AGX Orin	0.346	0.211	0.324	3.990
512	RTX3060 Ti	0.346	0.211	0.324	1.989
768	RTX3060	0.438	0.269	0.414	3.576
768	Jetson AGX Orin	0.438	0.269	0.414	8.270
768	RTX3060 Ti	0.438	0.269	0.414	3.334
1024	RTX3060	0.485	0.306	0.462	5.864
1024	Jetson AGX Orin	0.485	0.306	0.462	12.540
1024	RTX3060 Ti	0.485	0.306	0.462	5.376
1280	RTX3060	0.522	0.327	0.498	8.108
1280	Jetson AGX Orin	0.522	0.327	0.498	16.880
1280	RTX3060 Ti	0.522	0.327	0.498	7.955

Accuracy Fluctuations While Driving. As the vehicle navigates different environments, scenes can vary significantly: they may feature heavy traffic with numerous complexities or be much emptier. This diversity results in a wide range of object counts and complexities. Fig. 1 illustrates how model accuracy changes across different resolutions for all five cameras throughout a video. In simpler scenes, low-resolution models can effectively detect objects, whereas in more challenging scenes, high-resolution models are required for accurate detection.

Effect of Dynamically Changing Resolution. To justify the need for a dynamic scheduler, we compare two schedulers: a static scheduler, in which the resolution remains constant throughout the video, and an oracle scheduler that assumes the model's per-frame accuracy is known before execution. We evaluated five fixed resolutions, 256, 512, 768, 1024, and 1280, for the static scheduler. The Oracle scheduler provides a theoretical upper bound on the achievable latency and accuracy. It has a priori knowledge of how each model performs before inference and selects the lowest resolution for which $mAP \geq 0.7$. Table 3 shows the evaluation of static and oracle schedulers through 194 videos. Columns depict: *mAP*, *mAP*@50, and *mAP*@0.5-0.95 for accuracy; average, 95th percentile, and 100th percentile total latency for processing all frames from five cameras periodically; average percentage of frames passing 0.7 *mAP* threshold (pass rate %); and average difference in accuracy when *mAP* is below the 0.7 threshold. The Static-256 and Static-512 cases exhibit low latencies, but their pass rates are 40.92% and 50.61%, respectively. When they fall below the threshold, the p75 and p95 L1 distances indicate poor accuracy.

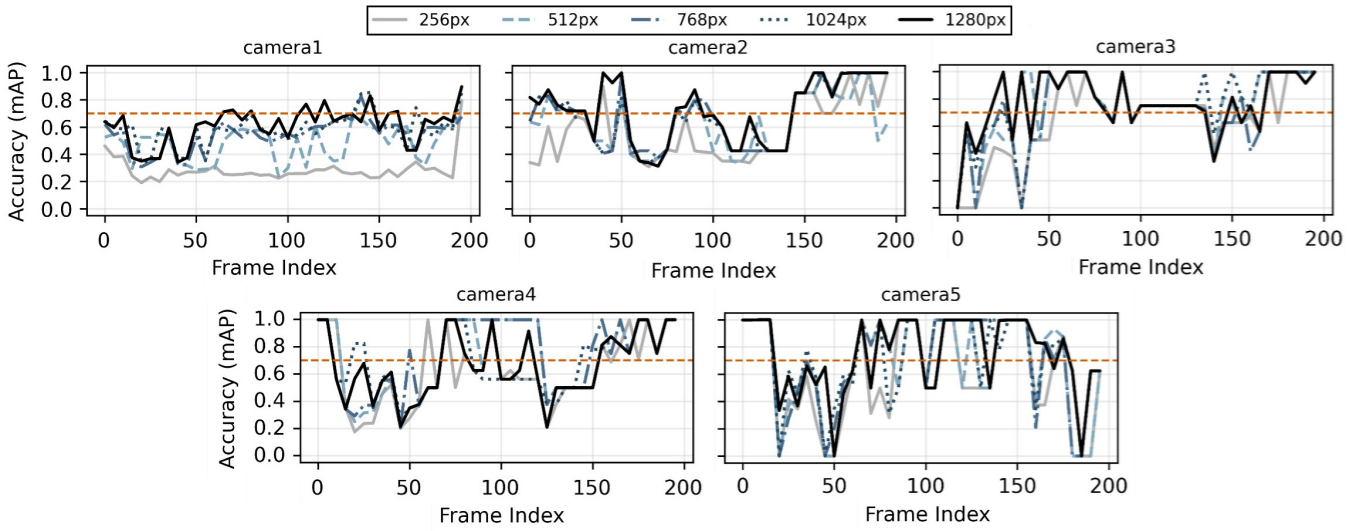


Figure 1: Per-frame *mAP* of Yolov9 using different resolutions for each camera over a video from the Waymo Dataset.

Table 3: Performance summary on Jetson AGX Orin.

Scheduler	*mAP*	*mAP* @50	*mAP* @50−95	Lat. (Avg.)	Lat. (p95)	Pass (%)	L_1 (Avg)	L_1 (p50)	L_1 (p75)	L_1 (p95)
Static-256	0.58	0.62	0.50	13.70	13.70	40.92	0.23	0.17	0.38	0.60
Static-512	0.67	0.71	0.58	20.50	20.50	50.61	0.15	0.08	0.24	0.52
Static-768	0.71	0.75	0.61	34.80	34.80	55.97	0.12	0.05	0.18	0.47
Static-1024	0.74	0.77	0.63	51.20	51.20	60.84	0.10	0.03	0.15	0.43
Static-1280	0.76	0.79	0.64	79.80	79.80	64.26	0.09	0.02	0.12	0.41
Oracle	0.76	0.80	0.62	36.03	49.69	68.48	0.07	0.01	0.09	0.35

4 Dynamic Scheduling

The performance of the oracle scheduler underscores the need to dynamically identify the resolution that ensures $mAP \geq 0.7$. This task has ordinal characteristics; therefore, we selected ordinal regression as our approach. Ordinal regression, which classifies instances into ordered categories [15], is commonly used in computer vision for tasks such as age and depth estimation, as well as for predicting image memorability and difficulty [5]. For these applications, classifiers like Convolutional Neural Networks (CNNs) and regression trees are often employed [20].

In CNNs, detection accuracy increases with increasing resolution. Our goal is to identify the lowest resolution that achieves an $mAP \geq 0.7$. Since CNNs are applied to each frame, the detected objects are already available. We trained an ordinal regression model (OGBoost library) using the features of the detected objects from the CNN to determine the optimal resolution that meets the $mAP \geq 0.7$ threshold [10]. We need to run this prediction for each frame because sudden changes can occur, as shown in the previous sections. Features can be sourced from any of the five resolutions at runtime. We trained five ordinal regression models, each corresponding to features extracted at a specific resolution. We select the appropriate model at runtime by matching it to the detection resolution from the previous frame. Fig. 2 illustrates our framework in action. An ordinal regression model prediction takes 1.05 ms on average on the CPU. Although per-camera overhead is 1.05 ms, since all five predictions are run in parallel, the total prediction overhead is 1.05 ms.

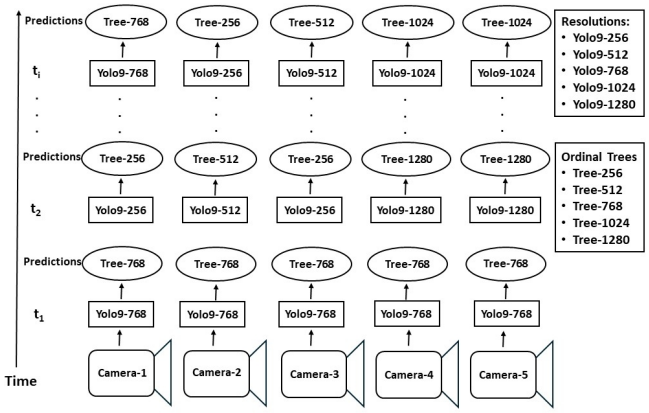


Figure 2: Illustration of dynamic 2D object detection scheduling in action. Camera frames are resized into different square resolutions and fed into the relevant YOLO network. After each inference, the trees' predictions are used to adjust the model configuration as needed to improve accuracy or reduce latency.

5 Experimental Results and Analysis

Workloads: After training YOLO networks with five different resolutions — 256, 512, 768, 1024, and 1280 — and an ordinal regression model, we evaluated the schedulers on 194 test videos with varying levels of difficulty. See Section 3 for the dataset specifications.
Hardware Platform: We used three hardware platforms: the NVIDIA RTX 3060, NVIDIA RTX 3060 Ti, and Jetson AGX Orin. Their characteristics are as follows.

RTX 3060: 28 SMs (3584 CUDA cores), 112 Tensor Cores, 12.7 TFLOPs (FP16), 170 W TDP; paired with Intel Core i7-13700K (16 cores, 0.8–5.4 GHz, 9.9 W idle).

RTX 3060 Ti: 28 SMs (3116 CUDA cores), 112 Tensor Cores, 12.7 TFLOPs (FP16), 170 W TDP; paired with Intel Core i7-12700KF (20 cores, 0.8–5.4 GHz, 32.9 W idle).

NVIDIA Jetson AGX Orin: Embedded Ampere SoM with 2048 CUDA and 64 Tensor Cores; 12-core Cortex-A78AE CPU (2.2 GHz max, 3 MB L2 + 6 MB L3), tested in maximum-performance mode.

The RTX series is more powerful than the Jetson AGX Orin. Since all models use TensorRT formats, model accuracy remains consistent across platforms for a given resolution. We used the Ultralytics library [12] to train and export models. The average latencies from Table 2 were used to calculate the scheduler performance.

Per-Camera Analysis. Fig. 3 presents distributions of per-camera mean average precision (*mAP*) values across schedulers. The front camera, Camera 1, has lower accuracy than the other cameras, suggesting that it receives more complex frames. The Oracle and dynamic schedulers enhance front-camera accuracy, outperforming static schedulers at resolutions of 768, 512, and 256. The leftmost bar in each group indicates that static schedulers with resolutions of 256, 512, or 768 may yield low accuracy on the front camera in some videos.

Fig. 4 shows the overall and per-camera latencies of the schedulers for the test videos on Jetson AGX Orin. The static schedulers use the same resolution throughout the videos, so latency remains constant. The dynamic scheduler achieves lower mean latency and comparable p95 latency to Static-1024, and lower mean and p95 latency than Static-1280. In some cases, it can even outperform the Static-768 scheduler. Camera 1 typically requires higher per-camera latency, whereas the other cameras typically use low-resolution models in Oracle and dynamic schedulers. The dynamic scheduler can reduce latency for these cameras and switch to high-resolution models as needed. Although latencies on the RTX3060 and RTX3060Ti architectures are lower than those on the Jetson AGX Orin, they exhibit similar patterns across schedulers. Appendix A presents a detailed analysis of accuracy and latency across video difficulty levels.

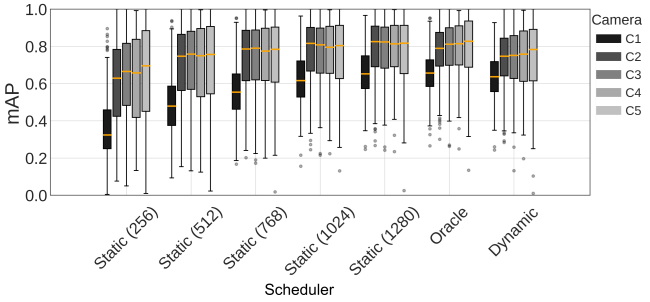
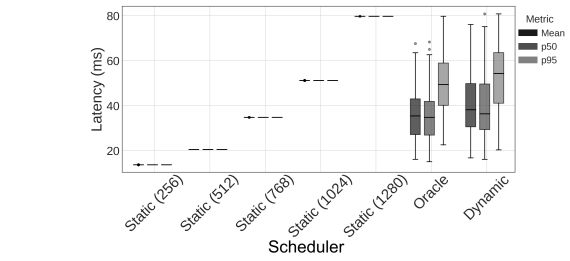


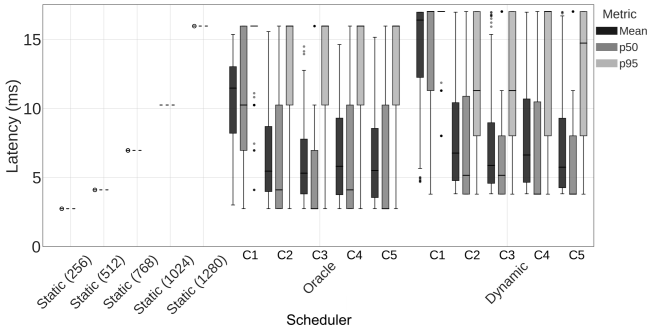
Figure 3: Per-camera accuracy comparison of schedulers.

6 Conclusions and Future Work

We aim to improve latency and per-frame accuracy for 2D object detection. Previous work mostly relied on large models to maintain high accuracy, at the expense of high latency and energy consumption, or on cloud offloading schemes. The former increases power consumption, while the latter is not always feasible in real time due to network constraints. We cannot achieve the desired accuracy while meeting latency constraints with a single resolution across all frames without a high-end processing unit. We addressed these issues using a dynamic approach. We dynamically adjust the input resolution in a single-processor setup to minimize latency while maintaining acceptable accuracy. We simulated static, Oracle, and dynamic schedulers with five different resolutions. The dynamic scheduler employs an ordinal regression model that utilizes the image and object features of the previous frame. Our experiments



(a) Overall



(b) Per-camera

Figure 4: Overall and per-camera latency distributions of schedulers on Jetson AGX Orin. The static schedulers are depicted as dashed lines, as the resolution remains constant.

demonstrated that a dynamic scheduler reduces latency and provides faster reaction times for easy videos with minimal accuracy loss, while maintaining higher accuracy than static, low-resolution schedulers on challenging videos. Our approach treated each camera as an independent data source during scheduling. Information from multiple cameras could be fused before inference to improve scheduling decisions.

Appendix A Analysis Based on Difficulty Levels

Fig. 5 shows the latency, pass rate, and L1 error for the schedulers at three difficulty levels. For easy sequences, the dynamic scheduler provides pass rates comparable to those of the Static-768 and Static-1024 schedulers. The mean latency of the dynamic scheduler is lower than Static-1024 in ~75% of cases. The dynamic scheduler also achieves lower 95th-percentile latency in more than 75% of cases than Static-1024. The L1 error is low in easy videos, so choosing the Dynamic scheduler yields lower latency than the high-resolution static schedulers while giving negligible accuracy differences.

The dynamic scheduler achieves a pass rate comparable to Static-1024 for medium-difficulty videos, though it sometimes shows a slightly lower pass rate. In contrast, the Static-768 scheduler can drop to 30% accuracy in ~25% of videos, whereas the dynamic scheduler consistently maintains an accuracy of at least 38%. Its L1 error matches that of Static-768, but can also outperform it in some cases. The dynamic scheduler's mean latency is better in ~75% of cases, and its p95 latency is better in ~50% of cases, compared to Static-1024. Overall, the dynamic scheduler achieves competitive accuracy and slightly lower latency than the Static-1024 scheduler.

For hard videos, even the Oracle scheduler can achieve a pass rate of 60% or less, indicating a significant L1 error rate in the analysis. The Dynamic scheduler's pass rates and L1 errors are comparable to those of the Static-1024 and Static-1280 schedulers.

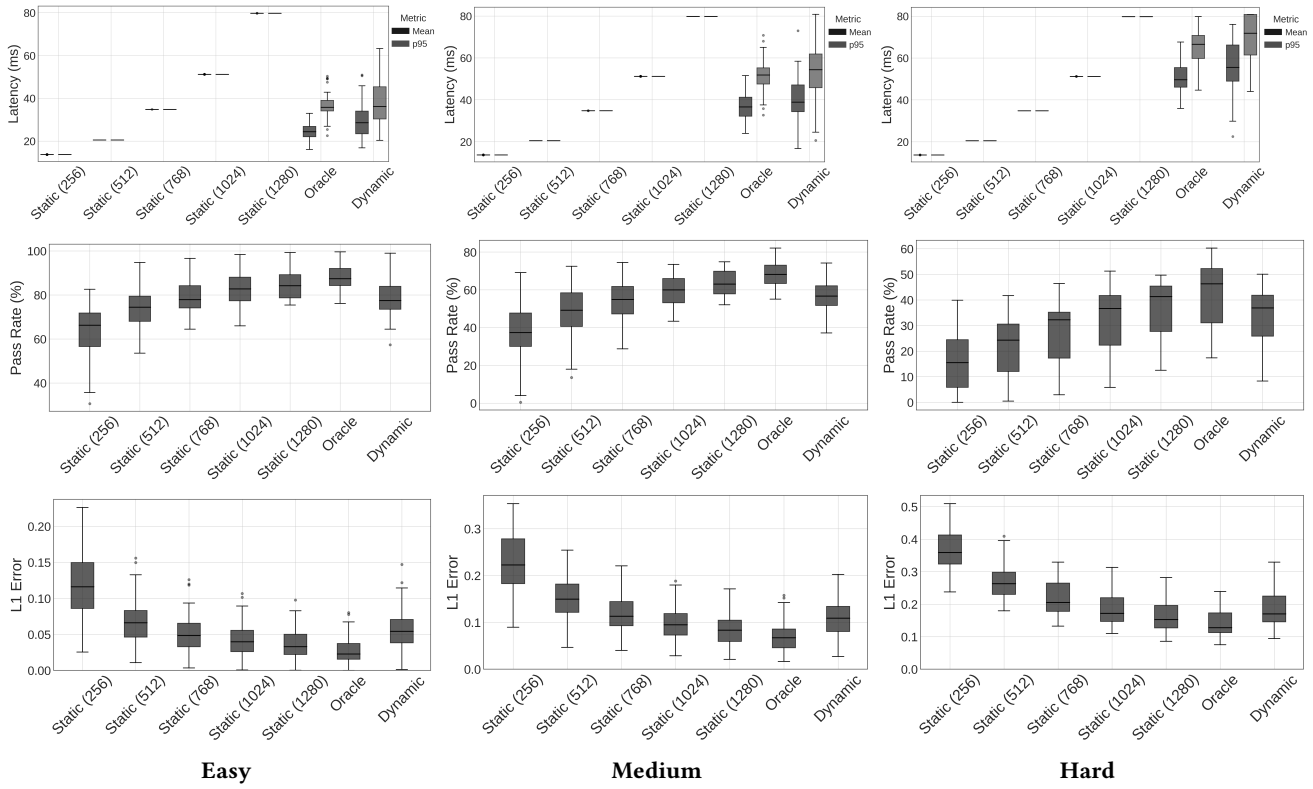


Figure 5: Distributions showing the performance of schedulers across difficulty levels. Top row: Latency distributions (Jetson). Middle row: Pass-rate distributions. Bottom row: L1 error rate distributions.

In terms of latency, the dynamic scheduler achieves a latency comparable to that of the Static-1024 and Static-1280 schedulers. Our regression model leverages features from detection models, yet even the highest-resolution models sometimes struggle to detect objects in challenging videos. Providing object-level features, such as the number of objects and their areas, can be misleading, causing the regressor to select a lower resolution than necessary for the next frame. Additionally, sudden changes between consecutive frames can lead to incorrect predictions when the dynamic scheduler relies on the previous frame's inference results.

References

[1] Autoware. 2023. Autoware Documentation. https://autowarefoundation.github.io/autoware-documentation/main/ Accessed: 2026-02-28.

[2] Google Coral Team. 2020. Coral USB Accelerator. https://coral.ai/products/accelerator/ Accessed: 2026-02-28.

[3] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. 2019. A Survey of Deep Learning Techniques for Autonomous Driving. *Journal of Field Robotics* 37, 3 (2019), 362–386. doi:10.1002/rob.21918

[4] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E. Haque, Lingjia Tang, and Jason Mars. 2018. The Architectural Implications of Autonomous Driving: Constraints and Acceleration. *SIGPLAN Notices* 53, 2 (2018), 751–766.

[5] David Mayo, Jesse Cummings, Xinyu Lin, Dan Gutfreund, Boris Katz, and Andrei Barbu. 2023. How hard are computer vision datasets? calibrating dataset difficulty to viewing time. *Advances in Neural Information Processing Systems* 36 (2023), 11008–11036.

[6] NVIDIA Corporation. 2022. *NVIDIA Jetson AGX Orin Series: Technical Brief.* Technical Report. NVIDIA. https://www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf Accessed: 2026-02-28.

[7] NVIDIA Corporation. 2024. Compare GeForce Graphics Cards. https://www.nvidia.com/en-us/geforce/graphics-cards/compare/ Accessed: 2026-02-28.

[8] NVIDIA Corporation. 2025. NVIDIA DRIVE Solutions for Autonomous Vehicles. https://developer.nvidia.com/drive Accessed: 2026-02-28.

[9] K. Ramamritham and J.A. Stankovic. 1994. Scheduling Algorithms and Operating Systems Support for Real-time Systems. *Proc. IEEE* 82, 1 (1994), 55–67. doi:10.1109/5.259426

[10] Mansour T. A. Sharabiani, Alex Bottle, and Alireza S. Mahani. 2025. OGBoost: A Python Package for Ordinal Gradient Boosting. arXiv:2502.13456 [stat.CO] https://arxiv.org/abs/2502.13456

[11] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, et al. 2020. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '20)*. IEEE, Piscataway, NJ, USA, 2443–2451. doi:10.1109/CVPR42600.2020.00252

[12] Ultralytics Inc. 2025. Ultralytics Library. https://docs.ultralytics.com/ Accessed: 2026-02-28.

[13] Ultralytics, Inc. 2025. YOLO Performance Metrics. https://docs.ultralytics.com/guides/yolo-performance-metrics/ Accessed: 2026-02-28.

[14] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. 2025. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. In *Computer Vision – ECCV 2024*. Springer Nature Switzerland, Cham, 1–21.

[15] Jinhong Wang, Jintai Chen, Jian Liu, Dongqi Tang, Danny Z. Chen, and Jian Wu. 2025. A Survey on Ordinal Regression: Applications, Advances and Prospects. arXiv:2503.00952 [cs.CV] https://arxiv.org/abs/2503.00952

[16] Waymo LLC. 2025. Waymo Autonomous Driving Technology. https://waymo.com. Accessed: 2026-02-28.

[17] Xilinx Inc. 2025. ZCU102 Evaluation Kit. https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html Accessed: 2026-02-28.

[18] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. 2021. Center-based 3D Object Detection and Tracking. arXiv:2006.11275 [cs.CV] https://arxiv.org/abs/2006.11275 Accessed: 2026-02-28.

[19] Yueming Zhang, Xiaolin Song, Bing Bai, Tengfei Xing, Chao Liu, Xin Gao, Zhihui Wang, Yawei Wen, Haojin Liao, Guoshan Zhang, and Pengfei Xu. 2021. 2nd Place Solution for Waymo Open Dataset Challenge – Real-time 2D Object Detection. arXiv:2106.08713 [cs.CV] https://arxiv.org/abs/2106.08713

[20] Haiping Zhu, Hongming Shan, Yuheng Zhang, Lingfu Che, Xiaoyang Xu, Junping Zhang, Jianbo Shi, and Fei-Yue Wang. 2021. Convolutional ordinal regression forest for image ordinal estimation. *IEEE Transactions on Neural Networks and Learning Systems* 33, 8 (2021), 4084–4095.