

# Multi-Resolution End-to-End Deep Neural Network for Optimizing Latency-Accuracy Tradeoff in Autonomous Driving

Qitao Weng and Heechul Yun  
University of Kansas  
Lawrence, KS, USA  
{wengqt, heechul.yun}@ku.edu

**Abstract**—Latency–accuracy tradeoffs are fundamental in real-time applications of deep neural networks (DNNs) for cyber-physical systems. In autonomous driving, in particular, safety depends on both prediction quality and the end-to-end delay from sensing to actuation. We observe that (1) when latency is accounted for, the latency-optimal network configuration varies with scene context and compute availability; and (2) a single fixed-resolution model becomes suboptimal as conditions change.

We present a multi-resolution, end-to-end deep neural network for the CARLA urban driving challenge using monocular camera input. Our approach employs a convolutional neural network (CNN) that supports multiple input resolutions through per-resolution batch normalization, enabling runtime selection of an ideal input scale under a latency budget, as well as resolution retargeting, which allows multi-resolution training without access to the original training dataset.

We implement and evaluate our multi-resolution end-to-end CNN in CARLA to explore the latency–safety frontier. Results show consistent improvements in per-route safety metrics—lane invasions, red-light infractions, and collisions—relative to fixed-resolution baselines.

## I. INTRODUCTION

End-to-end (E2E) deep-learning–based autonomous driving establishes a cyber-physical control loop in which a deep neural network (DNN), operating on sensor data, directly commands a physical vehicle via actuators. Safety in this loop depends jointly on prediction quality and end-to-end latency: any delay between sensing and actuation causes commands to operate on stale information. In urban autonomous driving settings, perception demands vary substantially along the route: lane-following segments tolerate coarser spatial detail, while intersections require reliable detection of small, safety-critical cues such as traffic lights and cross traffic. Because convolutional inference cost grows with input resolution, accuracy and latency are inherently coupled, motivating approaches that explicitly reason about both.

DNN-based E2E policies are increasingly adopted in many cyber-physical systems (CPS) because they directly optimize the final control objective—jointly tuning perception, prediction, and control—rather than relying on human-defined intermediate targets in modular stacks [1], [2]. Compared with modular pipelines, E2E models offer a unified trainable architecture, align all intermediate representations with the task,

improve computational efficiency through shared backbones, and scale effectively with additional data and compute [3].

Prior E2E driving policies [4], [1] typically fix input resolution and compute, optimizing accuracy while leaving loop-level latency implicit. Some recent studies investigate the latency and accuracy tradeoffs in E2E systems [5], [6]. Other work characterizes these tradeoffs within perception modules, but outside closed-loop control [7]. Crucially, none of these prior works support run-time selection of different resolutions using a single model. While D<sup>3</sup> [8] explores runtime switching between multiple pre-trained perception models based on environmental conditions, this requires maintaining several models in memory, limiting its applicability in resource-constrained embedded systems.

To address this gap, we leverage dynamic-resolution networks, which—through techniques such as resolution-aware batch normalization—enable a single backbone to efficiently support multiple input resolutions for image classification tasks [9], [10]. Building on this principle, we propose a multi-resolution end-to-end model for autonomous driving, specifically tailored to urban environments.

We instantiate a pre-trained, fixed-resolution single-image policy [11] and augment it with a multi-resolution backbone using shared convolutions and per-resolution batch normalization. This design maintains resolution-specific statistics within a single network, enabling operation at multiple input resolutions and facilitating runtime scale selection under a dynamic latency budget. Furthermore, we show that this pre-trained policy can be efficiently retargeted to new input sizes—without access to the original training data—via backbone-only fine-tuning while preserving the driving control head.

We make the following contributions:

- We empirically study the joint latency–accuracy–safety tradeoff for urban E2E driving, highlighting environment-dependent needs (e.g., intersections vs. lane following).
- We present a single-model, multi-resolution E2E architecture that supports multiple input scales and allows runtime scale selection without duplicating networks.
- We demonstrate resolution retargeting to new input sizes without access to the original training set, preserving policy behavior while enabling operation under different latency budgets.

These results collectively show that resolution adaptivity is a practical lever for improving the latency–safety frontier relative to fixed-resolution baselines.

## II. BACKGROUND AND MOTIVATION

1) *End-to-end Deep Neural Networks for Autonomous Vehicles*: Unlike standard robotics control approaches that decompose the task into multiple subtasks—each responsible for a specific, predefined function—the end-to-end (E2E) deep learning approach uses a single deep neural network to produce control outputs directly from raw sensor data. This dramatically simplifies the control pipeline [12] and has been explored for decades [4], [1], [13], [6].

Fig. 1 shows a high-level overview of a typical end-to-end system, composed of sensors, a DNN, and control actuators.

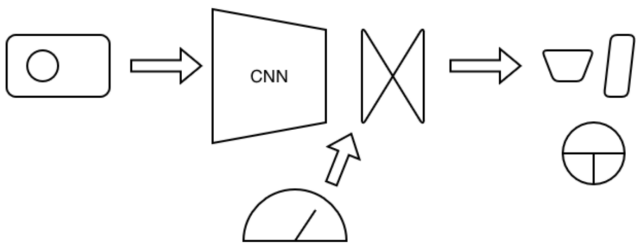


Fig. 1. End-to-end policy overview. A CNN backbone encodes the camera stream; ego-speed is provided as a lightweight proprioceptive input; a policy head (MLP) produces vehicle control commands.

2) *Importance of Latency in Autonomous Driving*: In an E2E autonomous driving system, a software policy, a DNN, observes the system status via sensors and directly issues low-level steering/throttle/brake commands. Safety of such a system depends on *accuracy*—the ability to perceive and react to small, critical cues (traffic lights, lane markings, curbs, pedestrians)—and on *latency*, the delay  $L$  between sensing and actuation.

This is because each command at time  $t$  is computed from measurements taken  $L$  seconds earlier, the vehicle has moved  $vL$  meters by the time the command takes effect (at speed  $v$ ), so the controller operates on stale information. Furthermore, the action is computed for an earlier world state and applied to a later one, introducing input-output phase lag and a state mismatch (approximately  $\Delta s \approx vL$ ,  $\Delta \theta \approx \omega L$  under a kinematic model with speed  $v$  and yaw rate  $\omega$ ). This latency-induced displacement also contributes a term  $vL$  to the stopping distance (with maximum deceleration  $a_{\max}$ ) [14],

$$d_{\text{stop}} = \frac{v^2}{2a_{\max}} + vL.$$

As  $L$  grows or fluctuates, stability margins shrink and tracking errors grow. Optimizing accuracy alone is therefore insufficient; *latency is a primary design constraint* that must be specified, measured, and enforced. Thus, accuracy and latency jointly determine closed-loop safety.

3) *Resolution and Latency Tradeoff*: Convolutional neural networks (CNNs) apply local filters over the spatial field; the dominant compute and memory costs scale with input area ( $\Theta(H \cdot W)$  for fixed channel widths). Higher input resolution typically improves recognition of small objects and fine structure but increases inference latency and memory pressure. Conversely, lower resolution reduces inference latency but risks losing critical cues at urban intersections. Thus, input resolution directly trades perceptual fidelity for end-to-end responsiveness.

4) *Implications*: Urban routes combine relatively easy *lane-following* segments, where coarse detail often suffices, with *intersections*, where small-object perception is essential. Lane-following is also typically executed at higher speeds, so a given latency  $L$  induces a larger state displacement  $\Delta s = vL$  (and angular error  $\Delta \theta \approx \omega L$ ) and a larger contribution  $vL$  to the stopping distance  $d_{\text{stop}}$ . Because the latency–accuracy operating point that maximizes safety depends on both scene context and available compute, fixed-resolution policies are inherently suboptimal under changing conditions. This motivates architectures and training protocols that expose resolution as a controllable degree of freedom while respecting closed-loop latency budgets.

## III. MULTI-RESOLUTION END-TO-END DEEP NEURAL NETWORK FOR AUTONOMOUS DRIVING

We adopt a monocular camera–based E2E controller from *Learning to Drive from a World on Rails (WoR)* [11], which was trained on its *NoCrash* dataset. The network uses a ResNet-34 [15] backbone (ImageNet-pretrained) to encode the camera stream. Figure 2 shows its overall architecture.

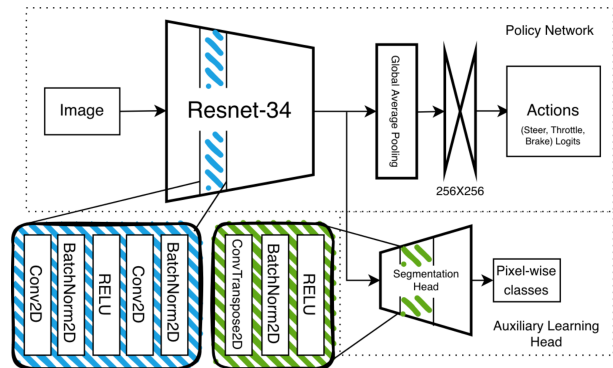


Fig. 2. Baseline network architecture

Our only modification to the policy network is to the convolutional backbone (ResNet-34), which we extend to support multiple input scales via per-resolution batch normalization, as described below. Note that the segmentation head is also extended to support multiple input scales, but it is only used during training for auxiliary learning as in the original setup.

### A. Per-Resolution Batch Normalization

We expose input scale as a controllable degree of freedom by sharing all convolutional weights and replicating *batch*

normalization (BN) per resolution. Let  $\mathcal{R} = \{r_1, \dots, r_m\}$  be the set of aspect-ratio-preserving scale factors relative to the WoR input size  $S_0 = (H_0, W_0)$ . For each backbone BN layer  $\text{BN}_\ell$  with  $C_\ell$  channels we create a bank

$$\{\text{BN}_\ell^{(r)} : r \in \mathcal{R}\}, \quad \text{BN}_\ell^{(r)} \equiv (\gamma_\ell^{(r)}, \beta_\ell^{(r)}, \mu_\ell^{(r)}, \sigma_\ell^{(r)}),$$

while keeping all convolutional weights  $\{W_\ell\}$  shared. Initialization copies the pretrained BN affine parameters and running statistics into every branch,  $\text{BN}_\ell^{(r)} \leftarrow \text{BN}_\ell$ .

At test time, exactly one BN branch per layer is active, matching the chosen input scale  $r$ ; this introduces no extra FLOPs beyond those implied by the input size. Parameter overhead is limited to the BN banks (per-scale  $\gamma, \beta$  and running  $\mu, \sigma$ ) and is negligible relative to convolutions.

1) *Multi-resolution training step*: At each optimizer step we process *all* scales in  $\mathcal{R}$  *sequentially*. For every  $r \in \mathcal{R}$ : (i) resize the entire micro-batch to  $(\lfloor rH_0 \rfloor, \lfloor rW_0 \rfloor)$  (no mixing of scales within a micro-batch), (ii) activate only the corresponding per-resolution batch-normalization branches  $\{\text{BN}_\ell^{(r)}\}$ , (iii) run a forward pass and accumulate that scale’s losses. We then average across  $r \in \mathcal{R}$  and perform a *single* backward/optimizer step, updating the shared convolutions and the active BN branch for each scale; the policy head remains frozen.

## B. Resolution Retargeting

1) *Data availability and provenance*: WoR [11] does not release the single-image *NoCrash* training set [11] used to train the baseline model. Therefore, we initialize from the WoR’s public single-image checkpoint and perform resolution retargeting using non-overlapping urban routes collected separately with the same sensor configuration and augmentation protocol.

2) *Protocol*: We adapt the CNN backbone to new input sizes while keeping the policy head fixed. We choose  $\mathcal{R} = \{0.75, 1.0\}$  as aspect-ratio-preserving scale factors relative to the WoR input size  $S_0$ . Training follows the multi-resolution schedule described above, with progressive unfreezing (layer4  $\rightarrow$  layer1).

3) *Optimization objective*: Let  $x$  denote the input image and  $c$  the route command. For a scale factor  $r \in \mathcal{R}$ , the network produces action-value logits  $z^{(r)}(x, c) \in \mathbb{R}^{|\mathcal{A}|}$  over a discrete action set  $\mathcal{A}$ . Following WoR [11], we form a target action distribution by applying a softmax (“Boltzmann policy”) to action-values supplied in the released dataset as ground-truth; denote this target by  $t(\cdot | x, c)$  with temperature  $\tau_0$ . The base supervision at scale  $r$  is the KL divergence from the target to the policy:

$$L_{\text{act}}^{(r)} = D_{\text{KL}}\left(t(\cdot | x, c) \parallel \text{softmax}(z^{(r)}(x, c))\right).$$

4) *Auxiliary segmentation*: Following WoR [11], we include semantic segmentation as an auxiliary objective and *train* the segmentation decoders jointly with the backbone. All batch normalization layers—including those in the segmentation branches—use per-resolution banks to avoid cross-scale statistic mismatch. The decoder is fully convolutional

and operates on spatial maps without global pooling, so batch-normalization moments  $\mu_c = \mathbb{E}_{b,h,w}[x_{b,c,h,w}]$  and  $\sigma_c^2 = \text{Var}_{b,h,w}[x_{b,c,h,w}]$  vary with input scale. For scale  $r$ , let  $s^{(r)}$  be the logits and  $y$  the ground-truth mask at label resolution. We resize logits to the label size (nearest neighbor) and use pixelwise cross-entropy:

$$L_{\text{seg}}^{(r)} = \text{CE}(\text{upsample}(s^{(r)}), y).$$

5) *Full-resolution teacher regularization*: To preserve the original policy while adapting to new input sizes, we distill from a *frozen* copy of the public checkpoint at full resolution (1.0 $\times$ ) as a teacher model. Let  $z^T(x, c)$  and  $f^T(x)$  be the teacher logits and pooled backbone features with dimensionality  $D$ , and  $f^{(r)}(x)$  the student features at scale  $r$ . We use:

$$L_{\text{KD}}^{(r)} = D_{\text{KL}}(\text{softmax}(z^T) \parallel \text{softmax}(z^{(r)})),$$

$$L_{\text{feat}}^{(r)} = \frac{1}{D} \|f^{(r)}(x) - f^T(x)\|_2^2.$$

6) *Aggregate retargeting loss*: Averaging over scales yields

$$L_{\text{retarget}} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \left( L_{\text{act}}^{(r)} + \lambda_{\text{seg}} L_{\text{seg}}^{(r)} + \lambda_{\text{KD}} L_{\text{KD}}^{(r)} + \lambda_{\text{feat}} L_{\text{feat}}^{(r)} \right),$$

with  $\lambda_{\text{seg}}, \lambda_{\text{KD}}, \lambda_{\text{feat}}$  set per experiment. The teacher is stop-gradient; student and teacher use identical data augmentations, differing only by the student’s isotropic resize to scale  $r$ .

## IV. EVALUATION

In this section, we present our evaluation setup and the results.

### A. Experimental Setup

1) *Simulator and timing*: All experiments run in the CARLA synchronous mode with the simulation tick at 40 Hz and camera sensors at 20 Hz. We evaluate three settings: (a) 50 ms period, no injected delay ( $D=0$ ); (b) 50 ms period with injected delay ( $D=50$  ms); and (c) latency-matched runs where we set  $D \in \{100, 150, 200\}$  ms and choose the controller period  $T_c=D$ . Injected delay is realized via zero-order hold: after a new command is computed, the previously applied command is held for  $D$  before the new command is latched. In synchronous CARLA, the achieved loop period is  $T_{\text{loop}} = \max(T_c, L_{\text{exec}})$ . When  $L_{\text{exec}} \leq T_c$ , the measured sensing-to-actuation delay is  $\approx D$ ; if  $L_{\text{exec}} > T_c$ , it is  $\approx L_{\text{exec}}$  (compute-bound).

2) *Routes and splits*: We evaluate on the two CARLA towns used by the WoR *NoCrash* policy: Town01 (seen during WoR training) and Town02 (unseen test). We reuse the PCLA codebase [16] to load the WoR single-image checkpoint as a CARLA agent and configure sensors and routes. Each town contributes *two* routes (four routes total). Unless noted, per-configuration summaries for a given town aggregate over its two routes and three traffic densities.

TABLE I  
SIMULATED TRAFFIC CONDITIONS USING CARLA’S RANDOMIZED SPAWNER.

Target (veh, ped)	Town01				Town02			
	Route 1		Route 2		Route 1		Route 2	
	Veh	Ped	Veh	Ped	Veh	Ped	Veh	Ped
(0, 0)	0	0	0	0	0	0	0	0
(5, 10)	5	7.0±1.5	5	7.0±1.5	5	6.6±1.4	5	6.6±1.4
(20, 40)	20	26.7±3.1	20	26.5±3.1	20	27.2±2.9	20	27.2±2.9

3) *Traffic workloads and trials*: Traffic is spawned by CARLA’s navigation mesh with randomized seeds, so realized counts can deviate from targets. We test three target densities  $(\#veh, \#ped) \in \{(0, 0), (5, 10), (20, 40)\}$ . To characterize this variability, for each (route, density) pair we perform 10 independent environment resets and record the realized vehicle/pedestrian counts; Table I reports the resulting mean±sd per route and town. For the main evaluation, we then run one rollout per (route, density) pair—two routes  $\times$  three densities per town—yielding 6 runs per town (12 total per configuration), and report performance metrics aggregated over these runs.

### B. Effects of Resolution Retargeting

In this experiment, we investigate the impact of resolution retargeting on driving performance of the evaluated models.

Table II reports success (route completion) rates and recoverable collision incidences (over 6 runs per town) of the Original WoR policy (Original), two fixed-resolution baselines (NRA 0.75×/1.0×), and our multi-resolution policy at 0.75×/1.0× (RA). The NRA baselines are models without BN banks for each resolution, but fine-tuned using the same method described in Subsection III-B. The number denotes the image-resolution scaling factor. Both NRA and RA configurations are resolution retargeted—i.e., fine-tuned for target resolutions on 50k frames from the 1M-frame CARLA Leaderboard dataset. Originally, all models were pre-trained on the 270k Town01 *NoCrash* dataset.

On Town01, all models complete the route without any collisions. On Town02, all models also successfully complete the route but resolution retargeted models (NRA and RA) experience 1 or 2 collisions, which they can recover to complete the route. From this, we find that resolution retargeting minimally impact performance despite being fine-tuned with a different dataset. Furthermore, we find that the two RA configurations with a single shared model perform similarly with the NRA baselines. In the following, we focus on comparing the performance of NRA and RA in more details.

### C. Effects of Input Resolution

In this experiment, we investigate the impact of input resolution on the performance of the evaluated models.

Fig. 3 shows the traffic-light infraction rates of the RA and NRA models under varying latency. As illustrated in the figure,

TABLE II  
DRIVING PERFORMANCE COMPARISON. NRA: NON-RESOLUTION-AWARE; RA: RESOLUTION-AWARE.

Map	Orig.	NRA 0.75×	NRA 1.0×	RA 0.75×	RA 1.0×
<i>Success (%)</i>					
Town01	100	100	100	100	100
Town02	100	100	100	100	100
<i>Collision incidence</i>					
Town01	0	0	0	0	0
Town02	0	1	1	1	2

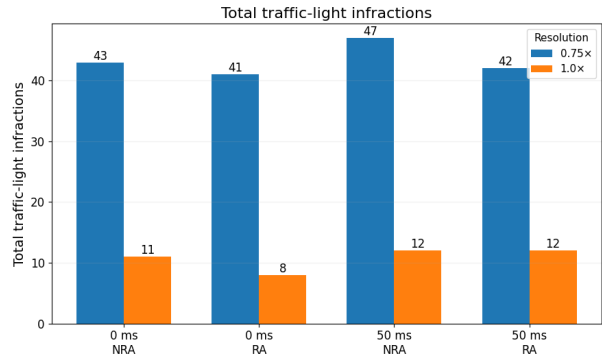


Fig. 3. Traffic-light infractions vs. input scale at a 50 ms control period, with/without 50 ms injected delay.

when the control period is fixed at 50 ms, reducing the input scale from the training-aligned 1.0× to 0.75× significantly increases traffic-light infractions across all configurations.

Fig. 4, on the other hand, shows the lane-invasion rates. Unlike traffic-light infractions, lane-invasion rates do not vary significantly with lower resolution. This is because traffic-light recognition relies on high resolution due to the small size of the lights, whereas lane detection is far less sensitive to reduced resolution.

These results show that input resolution materially influences urban E2E driving quality—particularly compliance with small-object cues—independent of the model variant.

### D. Effects of Latency

In this experiment, we investigate the impact of a model’s execution time (simulated by added latency) to its driving performance.

Fig. 5 shows the results. Across both NRA and RA models and at both scales, increasing injected delay produces a sharp transition from near-perfect operation to frequent catastrophic failure. With  $D \leq 100$  ms, success (route completion) rates remains high (91.65–100%) and collision rates are modest (0–25%). At  $D = 150$  ms, however, collision incidence jumps and success begins to drop significantly. By  $D = 200$  ms, success collapses to 0–8.35% and collisions reach 83–100% across configurations.

These results indicate that safety degrades as latency increases; moreover, the degradation is non-linear, with a “cliff” around  $\sim 150$  ms where success collapses and collisions spike.

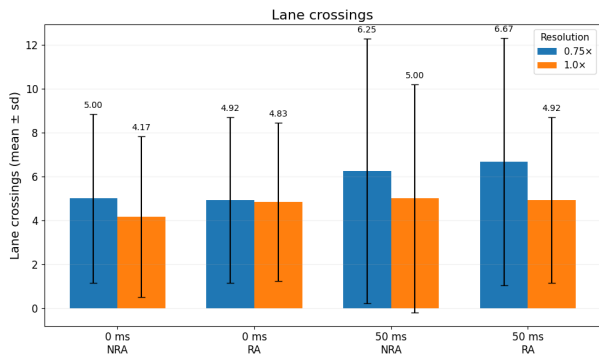


Fig. 4. Lane invasions per route (mean±sd) at a 50ms control period, with/without 50 ms injected delay.

### E. Runtime Latency-Accuracy Tradeoffs

In this experiment, we investigate the potential benefits of our approach by dynamically switching resolutions depending on the environment compared to the fixed resolution baselines.

Here, we assume an “oracle”, which decides which resolution is ideal for a given environment. That is, it triggers a resolution switch based on ego vehicle proximity to the traffic lights (default 0.75×; switch to 1.0× within 5 m of a traffic light; revert after exiting a 15 m radius). The intuition is that because higher resolution improve traffic light detection, such a policy can improve driving performance.

Table III shows the results. Note that the resolution-aware (RA) switcher outperforms both fixed baselines under the *same execution latency*. Under the 100→50 ms envelope, RA holds 100% success with 0% collisions, reduces lane invasions from  $6.67 \pm 5.63$  (fixed 0.75×) to  $5.50 \pm 5.21$  (~17.5% lower), and cuts red-light violations from 42 to 15 (~2.8× fewer). Under the tighter 150→100 ms envelope, RA again achieves 100% success and 0% collisions, lowers lane invasions from  $11.25 \pm 9.14$  (fixed 0.75×) to  $10.84 \pm 9.10$  (~3.6% lower), and reduces red lights from 43 to 17 (~2.5× fewer), significantly outperforming either of the fixed resolution baselines.

These results indicate that selectively spending compute near small-object bottlenecks (traffic lights) materially improves the latency-safety operating point compared to any fixed-scale policy with the same latency ceiling.

Note that the oracle trigger can be practically implemented by leveraging a map information as typically available in actual autonomous driving systems.

## V. RELATED WORK

1) *System-level latency-aware AV frameworks*: A line of work focuses on managing latency and accuracy at the *system* level in modular autonomous driving stacks. Pyrot is a modular AV platform built on a dataflow runtime that enables swapping perception and planning components and explicitly studying how their latency and accuracy affect end-to-end driving behavior in CARLA and on real vehicles [17]. Building on this stack, D<sup>3</sup> introduces a dynamic deadline-driven execution model and runtime that centralizes deadline

management and adapts the computation graph to changing driving context and resource availability, reducing collisions by adjusting module scheduling and quality under tight timing constraints [8]. These frameworks treat latency as a first-class concern at the pipeline level—deciding which modules to run and at what fidelity—but assume fixed-resolution perception networks trained separately and kept resident in memory at runtime—often impractical on embedded GPUs.

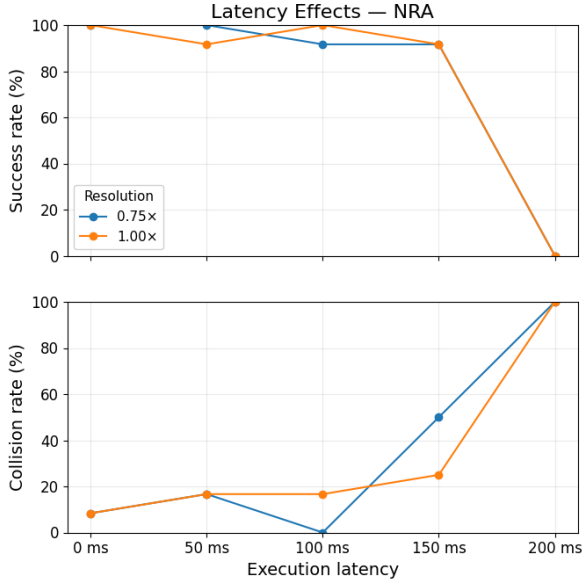
2) *Anytime perception modules for autonomous driving*: A complementary line of work develops perception components that trade accuracy for latency at run time, independent of the controller. In vision, *anytime* and budget-aware CNNs refine predictions progressively or use early exits to provide outputs under tight budgets; for example, anytime stereo networks generate intermediate disparity maps to enable latency-accuracy tradeoffs on embedded robots [18]. For object detection, dynamic and early-exit designs such as AdaDet [19] and AnytimeYOLO [20] adjust computation per frame while maintaining detection quality under inference-time limits. Others explored anytime processing of 3D LiDAR point clouds [21], [22], [23]. In particular, MURAL supports multi-resolution inference to meet per-frame deadlines [21]. These approaches demonstrate the value of deadline-aware, multi-resolution, or early-exit perception, but they remain within modular stacks and are evaluated on perception metrics, not closed-loop behavior. In contrast, we integrate multi-resolution capability directly into an end-to-end driving policy and evaluate its effect on urban driving safety under latency constraints.

## VI. CONCLUSION

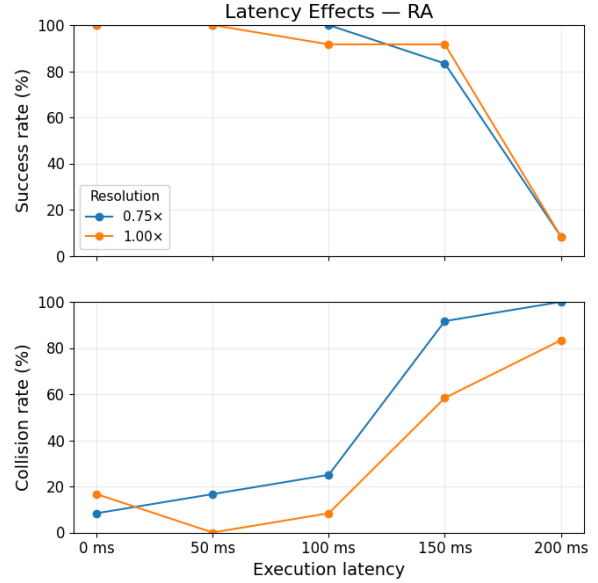
In this work, we studied how input resolution shapes the latency-accuracy trade space in closed-loop, end-to-end (E2E) urban driving. Building on a WoR single-image policy, we introduced a multi-resolution backbone with per-resolution batch normalization that (i) preserves baseline behavior, (ii) enables runtime selection of input scale under a latency budget, and (iii) supports resolution retargeting via backbone-only fine-tuning.

Two findings emerge across CARLA routes and traffic workloads. First, resolution strongly influences safety-critical behavior at a fixed control period: downscaling from the training-aligned size increases red-light violations and lane invasions, showing that small-object cues and geometric fidelity matter even when loop timing is constant. Second, latency shows a nonlinear “failure cliff”: as injected delay nears ~150ms, success collapses and collisions spike for all variants. Within these limits, we showed that a simple policy that switches to high resolution near intersections improves the latency-safety frontier relative to any fixed scale at the same latency ceiling.

Our resolution-aware backbone is lightweight (only BN banks are replicated), leaves the control head unchanged, and supports retargeting to new input sizes using non-overlapping routes, making resolution a practical runtime knob under compute constraints.



(a) NRA (non-resolution-aware)



(b) RA (resolution-aware)

Fig. 5. Success and collision rates vs. injected delay ( $D \in \{0, 50, 100, 150, 200\}$  ms) and input scale (0.75 $\times$ , 1.0 $\times$ ). Percentages are over 12 runs/configuration.

TABLE III  
RESOLUTION-AWARE (RA) SWITCHING VERSUS FIXED-RESOLUTION BASELINES UNDER THE SAME WORST-CASE EXECUTION LATENCY  $L_{\text{HIGH}}$ .

Latency envelope $L_{\text{high}} \rightarrow L_{\text{low}}$	Configuration	Runs	Success (%)	Collision (%)	Mean lane crossings	Red lights (total)
100 $\rightarrow$ 50 ms	Fixed 0.75 $\times$ @ 50 ms	12	100.00	16.65	$6.67 \pm 5.63$	42
	<b>RA switcher (50/100 ms)</b>	<b>12</b>	<b>100.00</b>	<b>0.00</b>	<b><math>5.50 \pm 5.21</math></b>	15
	Fixed 1.0 $\times$ @ 100 ms	12	91.65	8.35	$9.84 \pm 9.33$	<b>14</b>
150 $\rightarrow$ 100 ms	Fixed 0.75 $\times$ @ 100 ms	12	100.00	25.00	$11.25 \pm 9.14$	43
	<b>RA switcher (100/150 ms)</b>	<b>12</b>	<b>100.00</b>	<b>0.00</b>	<b><math>10.84 \pm 9.10</math></b>	<b>17</b>
	Fixed 1.0 $\times$ @ 150 ms	12	91.65	58.35	$66.50 \pm 76.01$	18

1) *Limitations and future work:* Our switching trigger is oracle-based and tested only in two simulated towns; broader generalization, hardware-in-the-loop timing, and jitter robustness remain open. Future directions include learned or map-based triggers, integration with explicit latency controllers, and real-vehicle evaluation under strict latency accounting.

#### ACKNOWLEDGMENT

This research is supported in part by NSF CPS-2038923 and Department of Defense (DoD) Contract number HQ00342310007.

#### REFERENCES

- [1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller *et al.*, “End to end learning for self-driving cars,” *arXiv:1604.07316*, 2016.
- [2] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, “End-to-end driving via conditional imitation learning,” in *ICRA*. IEEE, 2018.
- [3] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, “End-to-end autonomous driving: Challenges and frontiers,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [4] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” *Advances in neural information processing systems*, vol. 1, 1988.
- [5] A. Khalil and J. Kwon, “Plm-net: Perception latency mitigation network for vision-based lateral control of autonomous vehicles,” *arXiv:2407.16740*, 2024.
- [6] M. Bechtel, Q. Weng, and H. Yun, “Deeppicarmicro: Applying tinyml to autonomous cyber physical systems,” in *RTCSA*. IEEE, 2022.
- [7] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *CVPR*. IEEE, 2017.
- [8] I. Gog, S. Kalra, P. Schafhalter, J. E. Gonzalez, and I. Stoica, “D3: a dynamic deadline-driven approach for building autonomous vehicles,” in *EuroSys*. ACM, 2022.
- [9] Y. Wang, F. Sun, D. Li, and A. Yao, “Resolution switchable networks for runtime efficient image recognition,” in *ECCV*. Springer, 2020.
- [10] M. Zhu, K. Han, E. Wu, Q. Zhang, Y. Nie, Z. Lan, and Y. Wang, “Dynamic resolution network,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [11] D. Chen, V. Koltun, and P. Krähenbühl, “Learning to drive from a world on rails,” in *ICCV*. IEEE, 2021.
- [12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *Journal of Machine Learning Research*, vol. 17, no. 39, 2016.

- [13] M. G. Bechtel, E. McElhiney, M. Kim, and H. Yun, "Deeppicar: A low-cost deep neural network-based autonomous car," in *RTCSA*. IEEE, 2018.
- [14] D. Falanga, S. Kim, and D. Scaramuzza, "How fast is too fast? the role of perception latency in high-speed sense and avoid," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, 2019.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [16] M. J. Tehrani, J. Kim, and P. Tonella, "Pcla: A framework for testing autonomous agents in the carla simulator," in *FSE*, 2025.
- [17] I. Gog, S. Kalra, P. Schafhalter, M. A. Wright, J. E. Gonzalez, and I. Stoica, "Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles," in *ICRA*. IEEE, 2021.
- [18] Y. Wang, Z. Lai, G. Huang, B. H. Wang, L. Van Der Maaten, M. Campbell, and K. Q. Weinberger, "Anytime stereo image depth estimation on mobile devices," in *ICRA*. IEEE, 2019.
- [19] L. Yang, Z. Zheng, J. Wang, S. Song, G. Huang, and F. Li, "Adadet: An adaptive object detection system based on early-exit neural networks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 1, 2023.
- [20] D. Kuhse, H. Teper, S. Buschjäger, C.-Y. Wang, and J.-J. Chen, "You only look once at anytime (anytimeyolo): Analysis and optimization of early-exits for object-detection," *arXiv:2503.17497*, 2025.
- [21] A. Soyuyigit, S. Yao, and H. Yun, "Mural: A multi-resolution anytime framework for lidar object detection deep neural networks," in *RTCSA*. IEEE, 2025.
- [22] —, "Anytime-lidar: Deadline-aware 3d object detection," in *RTCSA*. IEEE, 2022.
- [23] —, "Valo: a versatile anytime framework for lidar-based object detection deep neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 11, 2024.