What Matters for Scalable and Robust Learning in End-to-End Driving Planners?

Supplementary Material

This supplementary material details our data scaling procedure (Sec. A), presents implementation details for our analysis framework and BevAD (Sec. B), and includes additional quantitative and qualitative results (Sec. C).

A. Dataset

A.1. Bench2Drive

Bench2Drive [28] offers expert demonstrations from the Think2Drive expert [33], comprising over 13,000 episodes for training. However, the adoption of the dataset within the community is limited due to missing sensor modalities and language annotations [37, 43, 56]. Furthermore, we identify significant 3D label flaws, as depicted in Fig. 6: (1) Unlabeled parked vehicles in all CARLA towns except Town12 and Town13, (2) Incomplete annotations for multi-lightbox traffic signals. (3) Misplaced bounding boxes for numerous traffic signs and pedestrians. These flaws create contradictory object detection supervision and enable planner shortcut learning, e.g., by distinguishing static cars from dynamic cars. Critically, the lack of public route files or open-source expert code hinder dataset extension and issue resolution. We therefore follow [37, 43, 56] and utilize the CARLA [13] Leaderboard 2.0 training routes and the rule-based PDM-lite expert [44, 56] for data collection.

A.2. Route Generator

Apart from the comprehensive Bench2Drive dataset, current CARLA imitation learning is limited by the diversity of expert demonstrations. SotA methods [37, 43, 56] utilize short, single-scenario route segments from CARLA Leaderboard 2.0 for training. However, this approach suffers from significantly imbalanced scenario distributions (e.g., scenario InterurbanAdvancedActorFlow appears only six times, while others appear 40 times). While these methods employ upsampling of rare routes and extensive camera- and weather augmentations, this mitigation is limited. Fixed geographic contexts and similar actor behaviors can lead to overfitting to spurious correlations [16].

To overcome these limitations, we build a novel route generator that creates unique route-scenario combinations within CARLA towns, enabling extensive and diverse data collection. Our approach maintains the established concept of short, single-scenario training routes, which facilitates fine-grained control over scenario distribution; for simplicity, we adopt a uniform distribution. The route generation algorithm proceeds in three steps: (1) Trigger Point Selection, (2) Route Planning and (3) Scenario Generation.

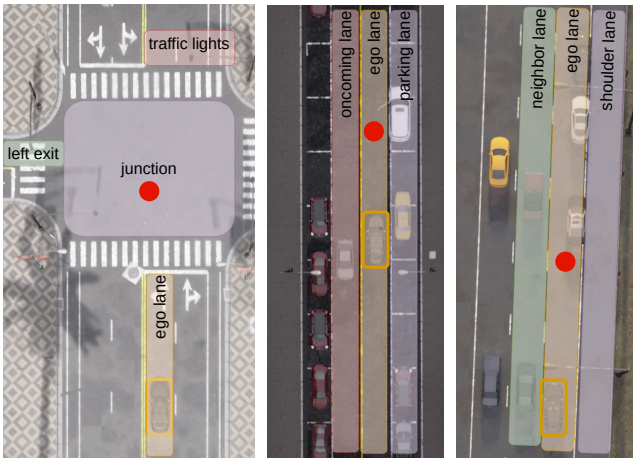
Figure 6. **Label Flaws.** Bench2Drive exhibits incomplete and erroneous 3D bounding box annotations in various scenes.

Trigger Point Selection. We employ a coarse-to-fine search strategy for trigger point selection. Initially, scenarios are mapped to one of three coarse location classes: Intersection, No-Intersection, or Highway Ramp. Subsequently, trigger points meeting these initial criteria are exhaustively sampled from all CARLA maps. This candidate list is then refined using scenario-specific criteria. For Intersections, relevant features include traffic lights, stop signs, turning options, bike lanes, and pedestrian crossings. For No-Intersections, we consider the number of adjacent lanes (with/against ego traffic flow), and the presence of parking or shoulder lanes. Highway Ramps are differentiated into On- and Off-Ramps.

For instance, the SignalizedJunctionLeftTurn scenario necessitates a signalized intersection with a left-turn option. The VehicleOpensDoorTwoWays scenario requires a right-side parking lane for the adversary and an adjacent lane with oncoming traffic. Conversely, the Accident scenario demands a right-side shoulder lane and a left-side lane with traffic flowing in the same direction. These examples are visualized in Fig. 7.

Route Planning. We plan routes through a trigger point using a bidirectional search on the lane graph to determine start and end points. This search is constrained to avoid additional intersections. Distances between the start point, trigger point, and end point are randomly sampled from scenario-dependent intervals. This strategy enhances variance and mitigates the learning of distance- and location-dependent shortcuts.

Scenario Generation. The final step involves configuring CARLA's pre-defined scenario behavior models at the trigger point. To enhance diversity and prevent spurious correlations, additional scenario parameters are sampled from meaningful intervals, for instance, by varying inter-vehicle distances within traffic flows.



(a) Signalized Junction Left (b) Open Door (c) Accident

Figure 7. **Visualization of Trigger Point Selection Criteria.** (a) This scenario requires a signalized junction with a left-turn exit relative to the ego lane. (b) The VehicleOpensDoorTwoWays scenario requires a right-side parking lane for the adversary and an adjacent left-side lane for oncoming traffic. (c) The Accident scenario demands a right-side shoulder lane for the group of blocking vehicles and an adjacent left-side lane with the same traffic flow.

Our route generator produces over 100,000 unique route-scenario combinations across all CARLA maps. Due to the high cost of sensor data collection, we utilize a subset of 8,000 routes for our data scaling experiments. Nevertheless, the generator's extensive diversity remains a significant asset for applications beyond imitation learning, such as reinforcement learning.

B. Implementation Details

B.1. Model

Efficient Streaming Training. The training performance of previous BEV-based E2E-AD architectures is bottlenecked by recurrent BEV feature generation, required for fusing temporal information within the BEV encoder's temporal self-attention layer [34]. For instance, UniAD [24] processes four past frames $(t-4,\ldots,t-1)$ at each training step t with a frozen BEV backbone to eventually compute temporal self-attention between $\mathcal{F}_{\text{Bev}}^{(t-1)}$ and $\mathcal{F}_{\text{Bev}}^{(t)}$. In contrast, we adapt streaming training from object-centric modeling [51] to BEV-based architectures. Specifically, during training, we sample streams of n subsequent frames, feeding them sequentially into the network. A memory component caches current BEV features $\mathcal{F}_{\text{Bev}}^{(t)}$ at each step t, enabling subsequent steps $(t' = t+1)$ to retrieve $\mathcal{F}_{\text{Bev}}^{(t'-1)} = \mathcal{F}_{\text{Bev}}^{(t)}$ from cache instead of recurrently recomputing it. Short sequences (e.g., 2s, $n = 20$ frames) are streamed to periodically reset the memory and mitigate non-orthogonal gradients [17] arising from strongly correlated frames. Our approach avoids four additional BEV backbone forward passes (compared to UniAD stage-1) and significantly reduces CPU load by requiring only one set of multi-view images per step, rather than five.

Method	BEV	FPS (train) ↑
UniAD-tiny [24, 28]	frozen (stage-2)	2.5
BevAD *(ours)*	frozen	89.0
	end-to-end	55.0

Table 5. **Training Efficiency.** BevAD achieves substantial training speed-up relative to UniAD-tiny. Measured on 8xA100-80GB.

Additionally, we employ RADIO [19] as an image backbone with a low-rank adapter (LoRA) [22] for parameter-efficient finetuning. RADIO provides rich, generic features, and LoRA finetuning significantly reduces VRAM requirements compared to backpropagating through a ResNet [18]. Pruning additional heads for online-mapping, motion forecasting, and occupancy prediction further reduces BevAD's memory footprint.

These optimizations combined enable end-to-end training with a batch size of 16 on a single A100-80GB GPU, a significant improvement over UniAD, which required two-stage training with a frozen backbone due to VRAM constraints. Our optimizations yield significantly higher training sample throughput, even surpassing UniAD-tiny, as shown in Tab. 5. Specifically, BevAD processes 35× more training samples per second with a frozen BEV backbone, and achieves a 22× speed-up in end-to-end training. These optimizations represent a major contribution towards scalable imitation learning for robust closed-loop performance, a benefit we extend to the community through our open-source code release.

Camera Augmentation. Camera augmentations are integral to robust closed-loop imitation learning [25]. Perfect expert drivers, such as PDM-lite, maintain precise lane centering, leading to a training distribution dominated by ideal states. However, during closed-loop testing, accumulated steering errors can cause the vehicle to drift, resulting in covariate shift [42] and degraded planner performance. A common mitigation involves augmenting driver camera views with random shifts and rotations during training [25], adopted by SotA methods on Bench2Drive [37, 43, 56]. This augmentation simulates out-of-distribution states not observed with perfect expert driving.

However, these shift and rotation augmentations are typically limited to single front-facing camera setups and are challenging to apply to multi-camera systems. Furthermore, their application to real-world data necessitates novel view synthesis, introducing pipeline overhead and potential artifacts. To address these limitations, we propose a

novel BEV-based augmentation strategy. Instead of manipulating raw sensor data, we augment the BEV coordinate system directly. This is achieved by sampling a random transformation matrix ${}^{Bev}\mathbf{T}_{Car}$, comprising a small yaw rotation $\gamma \sim [-22.5°, 22.5°]$ and a lateral offset $\Delta y \sim [-0.75m, 0.75m]$. This matrix is then applied to all camera transformation matrices ${}^{Car}\mathbf{T}_{\text{Cam}_i}$:

$$ {}^{\text{Bev}}\mathbf{T}_{\text{Cam}_i} = {}^{\text{Bev}}\mathbf{T}_{\text{Car}} \cdot {}^{Car}\mathbf{T}_{\text{Cam}_i} $$

By providing ${}^{\text{Bev}}\mathbf{T}_{\text{Cam}_i}$ to the BEV encoder, it builds BEV features $\mathcal{F}_{\text{Bev}}$ in the augmented BEV coordinate system rather than the vehicle coordinate system. Additionally, we apply this transformation to all ground truth labels, i.e., 3D bounding boxes and planning trajectories. Our augmentation prevents the detection head and planner from learning a bias towards axis-aligned objects or trajectories relative to the ego vehicle. For example, the disentangled planner must learn to predict the future path in the augmented BEV space, necessitating a robust understanding of the BEV feature grid. Unlike prior camera augmentations, our BEV-based scheme requires no augmented sensor data, as it only alters the generation of the latent BEV representation via a non-learnable, random transformation.

Planning Queries. The planning query's interpretation determines the output representation. For an entangled trajectory representation, we define $\mathcal{P}_{\text{Plan}} = \mathcal{P}_{\text{Traj}} \in \mathbb{R}^{N_t \times N_c}$, where N_t is the number of trajectory points and N_c is the feature dimension. For a disentangled representation, $\mathcal{P}_{\text{Plan}} = [\mathcal{P}_{\text{Path}} \parallel \mathcal{P}_{\text{Speed}}]$, with $\mathcal{P}_{\text{Path}} \in \mathbb{R}^{N_p \times N_c}$ and $\mathcal{P}_{\text{Speed}} \in \mathbb{R}^{N_t \times N_c}$, where N_p denotes the number of path points. In our experiments, we set $N_t = 15$ for a $3s$ planning horizon, resulting in temporal waypoint and speed predictions spaced at $0.2s$ intervals. For path planning, we use $N_p = 30$ with waypoints spaced $1m$ apart.

Planning Head. The planning head (Fig. 2b) comprises $N_{\text{layer}} = 8$ transformer decoder layers with a feature dimension of $N_c = 512$. Inspired by [41], we integrate an adaLN-Zero transformation to condition the self-attention and global cross-attention layers with $\mathbf{c}$, as depicted in Fig. 8. For the point estimator, $\mathbf{c} = \text{emb}(\text{command}) + \text{emb}(v_0)$, combining a learnable embedding of the high-level navigation command and a sinusoidal embedding of the current velocity. For the diffusion-based planner, $\mathbf{c} = \text{emb}(\text{command}) + \text{emb}(v_0) + \text{emb}(\tau)$, additionally incorporating an embedding of the current diffusion timestep τ.

B.2. Diffusion-based Planning

Preliminaries. Diffusion models are a class of generative models that learn to reverse a gradual noising process applied to data [20, 45]. They define a fixed forward Markov chain that progressively adds Gaussian noise to data, transforming it into a pure noise distribution. Specifically, the forward process admits sampling of noisy data $\mathbf{x}_\tau$ for arbitrary (time)steps τ in the Markov chain in closed-form given clean data $\mathbf{x}_0$ by sampling Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ [20]:

$$ \mathbf{x}_\tau(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha_\tau}}\mathbf{x_0} + \sqrt{1 - \bar{\alpha_\tau}}\boldsymbol{\epsilon} \tag{1} $$

The sequence $\bar{\alpha_0}, \dots \bar{\alpha_T}$ typically stems from a variance schedule with constant parameters [20, 46]. The core of diffusion models lies in learning a reverse Markov chain that iteratively denoises samples over T timesteps, starting from random noise $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$, to recover data samples from the original distribution [20]:

$$ p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{\tau=1}^{T} p_\theta(\mathbf{x}_{(\tau-1)}|\mathbf{x}_\tau) \tag{2} $$

Instead of learning the Gaussian transitions $p_\theta(\mathbf{x}_{\tau-1}|\mathbf{x}_\tau)$ directly, it is common practice to learn a function approximator $f_\theta : (\mathbf{x}_\tau, \tau) \to \mathbf{x}_0$ parameterized by a neural network with learnable weights θ [39]. This allows learning the diffusion model by minimizing Eq. (3) for all timesteps τ [20]:

$$ \|\mathbf{x}_0 - f_\theta(\mathbf{x}_\tau, \tau)\| \tag{3} $$

For diffusion-based planning, the clean planning ground truth $\mathbf{x}_0$ is defined based on the planning representation:

- Entangled: a series of normalized trajectory waypoints, $\mathbf{x}_0 = \{(x_i, y_i\}_{i=1}^{N_t}\}$.
- Disentangled: a tuple comprising normalized path waypoints and a normalized speed sequence, i.e., $\mathbf{x}_0 = (\{(x_i, y_i\}_{i=1}^{N_p}, \{(v_i\}_{i=1}^{N_t}\})$.

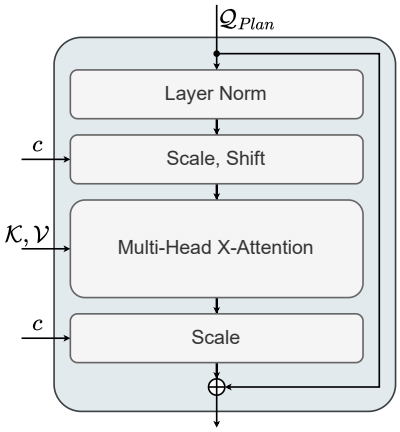


Figure 8. **adaLN-Zero Attention.** We employ an adaLN-Zero transformation to condition the transformer on $\mathbf{c}$. For self-attention, we set $\mathcal{K} = \mathcal{V} = \mathcal{Q}_{\text{Plan}}$ and for cross-attention, we set $\mathcal{K} = \mathcal{V} = \mathcal{F}_{\text{Scene}}$.

Training. During training, we sample a timestep τ from a uniform distribution and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. We obtain the noisy sample $\mathbf{x}_\tau$ using Eq. (1) and following the DDIM variance schedule [46]. The planning head serves as the *conditional* function approximator $f_\theta : (\tilde{\mathbf{x}}_\tau, \tau, \mathbf{z})$ for the reverse process [36], tasked with predicting $\mathbf{x}_0$ from $\tilde{\mathbf{x}}_\tau$, the diffusion timestep τ, and conditioning context $\mathbf{z} = (\mathcal{F}_{\text{Scene}}, c)$. Note that $\mathbf{x}_\tau$ is embedded into a latent space via an MLP to obtain $\tilde{\mathbf{x}}_\tau$ before being fed to f_θ.

Inference. For inference, we start with a random sample $\mathbf{x}_T \sim \mathcal{N}(0, 1)$ and iteratively denoise it using the trained function approximator $f_\theta(\tilde{\mathbf{x}}_\tau, \tau, \mathbf{z})$ and the DDIM sampling algorithm [46]. This process progressively denoises $\mathbf{x}_T$ to yield the final clean prediction $\mathbf{x}_0$. We specifically employ DDIM's accelerated generation that conducts denoising with a subset of $S < T$ denoising steps to enhance computational efficiency during sampling.

DiffusionDrive. We do not adopt DiffusionDrive's [36] truncated diffusion schedule in our experiments. This decision stems from our observation, that DiffusionDrive's forward process adds noise to a fixed set of anchor trajectories, while the reverse process aims to predict ground truth trajectories, leading to an asymmetric reversal. This issue is also thoroughly discussed in concurrent work [37], which proposes a theoretically sound diffusion bridge formulation as a solution.

B.3. Controller

We adopt the disentangled PID controllers from Simlingo [43] for lateral and longitudinal control. All controller parameters are retained, with the exception of the steering proportional gain, which is slightly lowered to $P_{\text{steer}} = 1.8$ to reduce oversteering in our closed-loop agent.

B.4. Loss

The overall loss function for end-to-end training combines terms for 3D object detection and planning:

$$\mathcal{L} = \lambda_{\text{det}}\mathcal{L}_{\text{det}} + \lambda_{\text{plan}}\mathcal{L}_{\text{plan}} \tag{4}$$

The detection loss $\mathcal{L}_{\text{det}}$, adopted from [34], includes classification and regression components. The planning loss $\mathcal{L}_{\text{plan}}$ varies with the planner's modeling choice:

- For regression-based planning with a point estimator, $\mathcal{L}_{\text{plan}}$ is a smooth L^1 loss applied to the trajectory error, or path and speed error.
- For diffusion-based planning, $\mathcal{L}_{\text{plan}}$ is a smooth L^1 loss on the $\mathbf{x}_0$-prediction error, as defined in Eq. (3).

The loss coefficients are set to $\lambda_{\text{det}} = 1$ and $\lambda_{\text{plan}} = 100$ to balance the magnitudes of the detection and planning loss components.

B.5. Metrics

To better characterize closed-loop failure modes, we introduce auxiliary metrics: the static infraction rate (IR_s) and dynamic infraction rate (IR_d).

$$\text{IR}_\text{s} = \frac{N_{\text{layout-collision}} + N_{\text{outside-lane}}}{N_{\text{routes}}} \tag{5}$$

This metric quantifies infractions related to lateral control errors, such as collisions with static layout elements or driving outside the lane.

$$\text{IR}_\text{d} = \frac{N_{\text{actor-collision}} + N_{\text{red-light}} + N_{\text{stop-sign}}}{N_{\text{routes}}} \tag{6}$$

This metric captures infractions arising from longitudinal control errors and interactions with dynamic elements, including collisions with actors, running red lights, or failing to stop at stop signs. Collectively, these metrics provide the expected number of infractions per route, offering a granular understanding of control deficiencies.

C. Results

This section presents an ablation study, quantitative results on the Bench2Drive benchmark, and qualitative demonstrations of BevAD's closed-loop driving.

C.1. Ablations

Our ablation studies investigate early design choices. For the following ablations of camera augmentation and BEV size, our baseline is a planning head with a tokenizer ($p = 4$, masking), disentangled representation, and a point-estimator regressor, as detailed in Sec. 3.3 and Tab. 1. Furthermore, we ablate the number of denoising steps in the diffusion planner using our strongest model, BevAD-M.

Camera Augmentation. To assess the effectiveness of our novel BEV-based augmentation, we compare the baseline against a variant trained without it. As shown in Tab. 6, the absence of camera augmentation significantly degrades closed-loop performance. This degradation primarily stems from a $5.7\times$ increase in static infractions, leading to reduced route completion and increased secondary collisions. These results underscore the critical role of our augmentation scheme in promoting robust driving and enabling recovery from compounding steering errors. The results also highlight the insufficiency of common open-loop metrics, as the L1 trajectory error does not reflect the observed degradation in model robustness.

BEV Size. Our tokenizer compresses high-resolution BEV features ($\mathcal{F}_{\text{Bev}}$) into low-resolution scene tokens ($\mathcal{F}_{\text{Scene}}$). An alternative is to directly learn a low-resolution BEV feature space. As shown in Tab. 7, despite exposing the same number of scene tokens to the planner, direct low-resolution BEV generation significantly degrades closed-loop performance. Specifically, the deformable BEV-image cross-attention of our BEV encoder (based on BEVFormer [34]) extracts image information more sparsely when generating

Augmentation	DS ↑	SR ↑	IR_s ↓	L1 (m) ↓
✓	**82.62**	**57.43**	**0.055**	**1.43**
✗	66.60	33.64	0.314	1.47

Table 6. **Ablation of camera augmentation.** The absence of camera augmentation significantly degrades closed-loop performance, despite minimal impact on open-loop L1 trajectory deviation. This underscores the contribution of our BEV-based augmentation to robust driving and covariate shift mitigation.

BEV	Scene Tokens	DS ↑	SR ↑	IR_s ↓
100×100	25×15	**82.62**	**57.43**	**0.055**
25×25	25×15	72.18	40.36	0.409

Table 7. **BEV Resolution and Tokenization.** High-resolution BEV generation, compressed via our tokenizer, yields superior closed-loop driving performance compared to direct low-resolution BEV generation.

S	DS ↑	SR ↑	FPS ↑
10	88.11	**72.73**	4.2
5	88.33	72.72	5.8
2	**88.53**	72.72	**7.5**

Table 8. **Impact of Denoising Iterations.** Reducing denoising steps significantly boosts inference FPS while preserving closed-loop driving performance, enabling real-time applications. FPS measured on a Quadro RTX 8000.

low-resolution BEVs, potentially omitting fine details. Furthermore, it prevents leveraging deformable refinement layers for sampling local, high-resolution features around the future trajectory. As a result, we observe $7.4\times$ more static infractions with the low-resolution BEV encoder compared to our compression approach. This finding underscores the necessity of compressing high-resolution representations rather than directly learning low-resolution ones.

Number of denoising steps. The iterative denoising of diffusion models critically impacts the inference latency in real-world deployments. The runtime of our diffusion-based planner linearly increases with the number of denoising steps S, while the point estimator planner has constant runtime, corresponding to $S = 1$. We thus evaluate how S affects closed-loop driving performance and inference FPS in Tab. 8. In contrast to DiffusionDrive [36], we achieve constant driving performance for $S \in \{2, 5, 10\}$, without applying their truncated diffusion framework. We attribute this to a bug in their diffusion schedule, which we detail in the appendix.

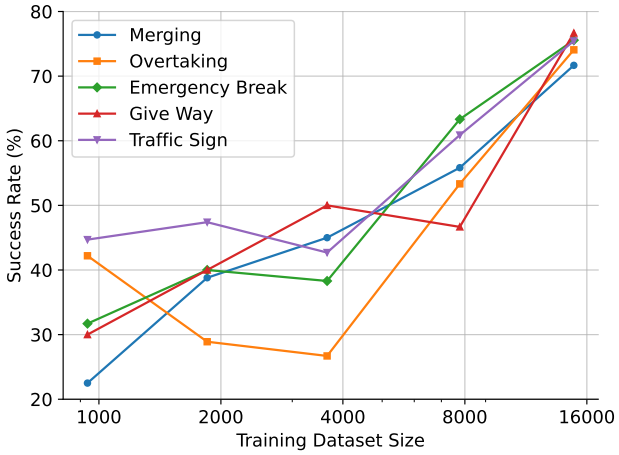


Figure 9. **Emerging Skills.** All skills remain underdeveloped with fewer than 4,000 training episodes. As the volume of training data increases, the skills progressively and uniformly emerge.

C.2. Multi-Ability Evaluation

To gain a nuanced understanding of system performance in closed-loop driving, we employ the fine-granular multi-ability evaluation protocol from Bench2Drive [28]. This protocol defines five advanced urban driving skills: Merging, Overtaking, Emergency Brake, Give Way, and Traffic Sign. Each of the 220 test routes is mapped to one or more skills necessary for successfully navigating the scenario. Tab. 9 presents a comparison of BevAD's multi-ability scores against prior work. BevAD consistently achieves high scores ($> 70\%$) across all skills, dominating the mean score. In contrast, prior state-of-the-art methods [43, 48] exhibit uneven performance, excelling in some skills while underperforming in others. BevAD-M surpasses the previous best in Merging and Give Way skills by +8.17 and +23.34, respectively. These skills demand comprehensive surround perception, underscoring BevAD's effective utilization of its multi-view camera system. However, BevAD lacks in terms of Overtaking, Emergency Brake, and Traffic Sign skills compared to the best prior or concurrent methods for each skill.

Building on the discussion in Sec. 3.5, we present the progression of BevAD's closed-loop driving skills as the training dataset size increases, detailed in Fig. 9. With fewer than 4,000 training episodes, all skills exhibit a success rate below 50%. However, as the training data is doubled and quadrupled, the skills show consistent and uniform improvement. This enhancement is evidenced by high success rates in complex scenarios, such as overtaking amidst oncoming traffic, merging onto highways and into traffic flow at intersections, and yielding to emergency vehicles.

Method	Ability (%) ↑					
	Merging	Overtaking	Emergency Brake	Give Way	Traffic Sign	**Mean**
VAD [29]	8.11	24.44	18.64	20.00	19.15	18.07
UniAD-Base [24]	14.10	17.78	21.67	10.00	14.21	15.55
ThinkTwice [27]	27.38	18.42	35.82	50.00	54.23	37.17
DriveAdapter [26]	28.82	26.38	48.76	50.00	56.43	42.08
Hydra-NeXt [35]	40.00	64.44	61.67	50.00	50.00	53.22
Orion [15]	25.00	71.11	78.33	30.00	69.15	54.72
TF++ [25]	58.75	57.77	83.33	40.00	82.11	64.39
Simlingo [43]	54.01 ±2.63	57.04 ±3.40	88.33 ±3.34	53.33 ±5.77	82.45 ±4.73	67.03 ±2.12
Hip-AD [48]	50.00	**84.44**	83.33	40.00	72.10	65.98
BridgeDrive† [37]	63.50	58.89	**88.34**	50.00	**88.95**	69.93
BevAD-S *(ours)*	55.83 ±0.72	53.33 ±6.67	63.33 ±1.93	46.67 ±5.77	60.88 ±3.08	56.01 ±3.78
BevAD-M *(ours)*	**71.67** ±2.60	74.07 ±1.29	75.56 ±4.41	**76.67** ±5.77	75.44 ±1.61	**74.68** ±1.24

Table 9. **Multi-Ability Evaluation.** BevAD consistenly achieves high results across all skills, dominating the mean score score. Notably, it significantly surpasses prior work in the *Merging* and *Give Way* skills. However, it exhibits comparatively lower performance in *Overtaking*, *Emergency Brake*, and *Traffic Sign* skills. *Legend:* †: concurrent work.

C.3. Qualitative Results

We provide qualitative closed-loop driving examples for each multi-ability skill of BevAD-M in Fig. 10, 11, 12, 13, 14 and 15. The examples are best viewed when zoomed in and visualize planned ego trajectories that are generated by rolling out the predicted speed profile along the predicted path. This depiction aids in understanding the dynamic planning component, but is solely for visualization. It does not serve as controller input, nor does it affect the lateral accuracy of the predicted path.

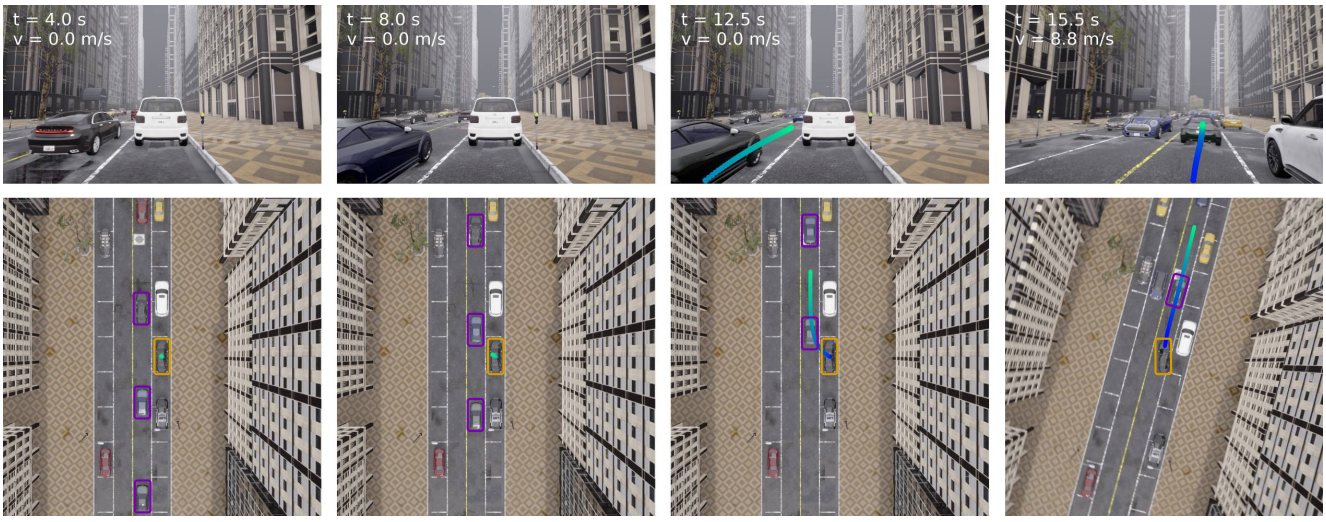


Figure 10. **Merging.** BevAD merges from a parallel parking space into traffic. It yields to rear-end flow of vehicles, identifies a safe gap, and accelerates for seamless merging.

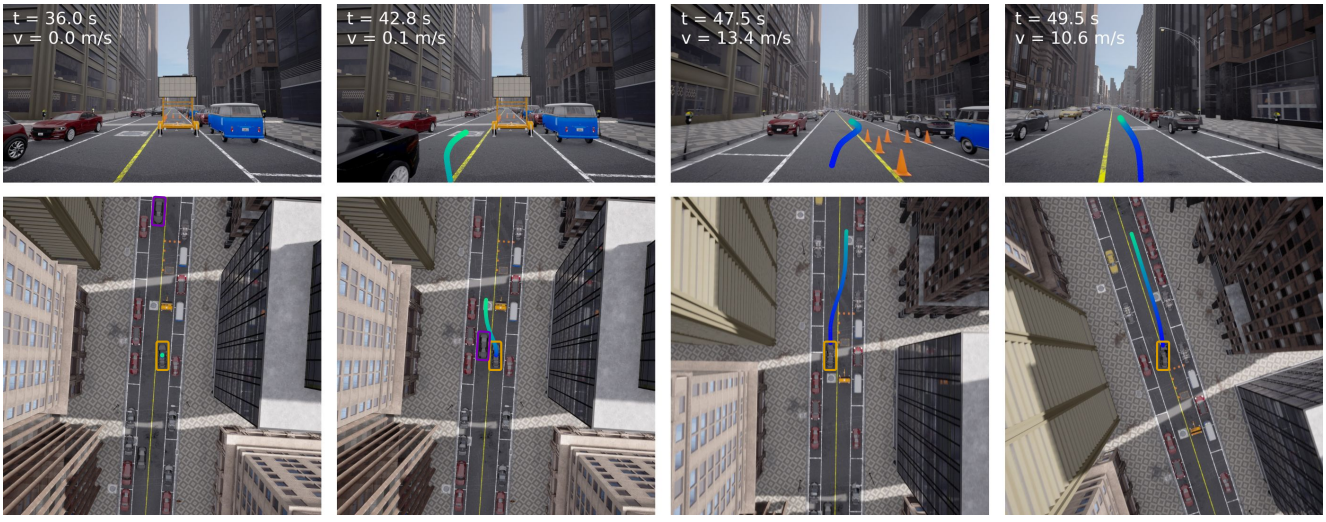


Figure 11. **Overtaking (1).** BevAD executes an overtaking maneuver when the route is blocked by a construction vehicle. It waits for clear oncoming traffic, then steers into the opposing lane, accelerates to quickly pass the obstacle, and subsequently decelerates when returning to its original lane.

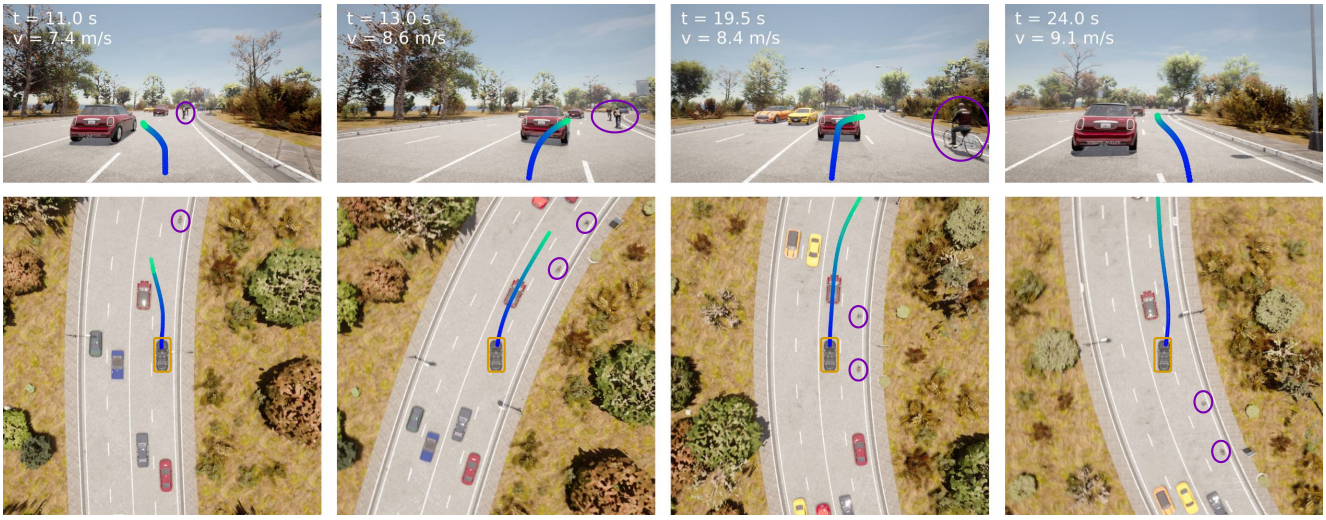


Figure 12. **Overtaking (2).** BevAD approaches a group of cyclists. It executes a safe left lane change to overtake them. After the maneuver, BevAD returns to its original lane, maintaining a safe distance to the cyclists.

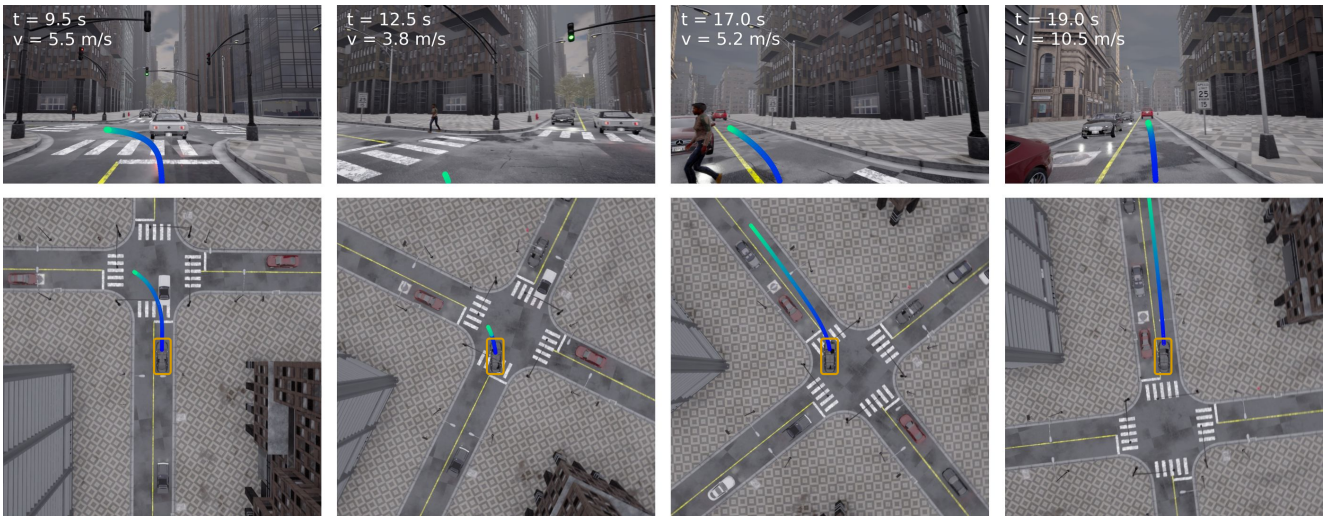


Figure 13. **Emergency Brake.** BevAD brakes at a green-light intersection due to a pedestrian crossing its left-turn path. Driving resumes upon pedestrian clearance.

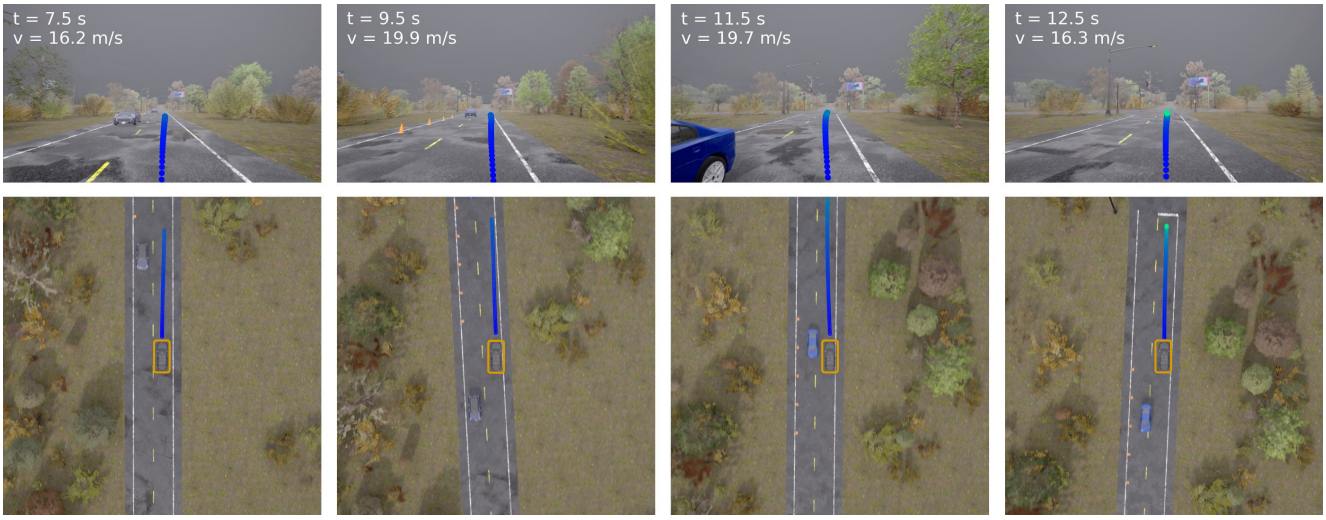


Figure 14. **Give Way.** BevAD yields to an oncoming vehicle encroaching on the ego lane. It performs a controlled rightward deviation from the lane center, staying within road limits, and re-centers once the oncoming traffic has passed.

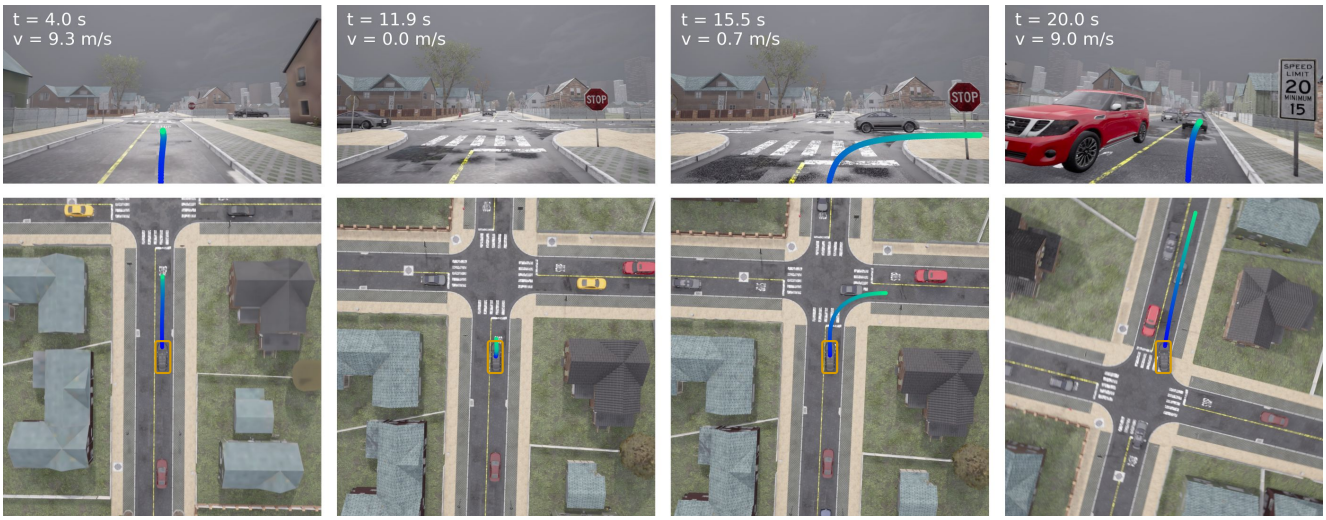


Figure 15. **Traffic Sign.** BevAD stops at a stop sign at an intersection with cross-traffic. It waits for a safe gap, then executes a right turn and merges into the traffic flow.