2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

CAPS: Context-Aware Priority Sampling for Enhanced Imitation Learning in Autonomous Driving

Hamidreza Mirkhani[1], Behzad Khamidehi[1], Ehsan Ahmadi[1,2], Mohammed Elmahgiubi[1], Weize Zhang[1], Fazel Arasteh[1], Umar Rajguru[1], Kasra Rezaee[1], Dongfeng Bai[1]
[1]*Noah's Ark Lab, Huawei Technologies Canada* [2]*University of Alberta*
Emails: firstname.lastname@huawei.com

***Abstract*—In this paper, we introduce Context-Aware Priority Sampling (CAPS), a novel method designed to enhance data efficiency in learning-based autonomous driving systems. CAPS addresses the challenge of imbalanced datasets in imitation learning by leveraging Vector Quantized Variational Autoencoders (VQ-VAEs). In this way, we can get structured and interpretable data representations, which help to reveal meaningful patterns in the data. These patterns are used to group the data into clusters, with each sample being assigned a cluster ID. The cluster IDs are then used to re-balance the dataset, ensuring that rare yet valuable samples receive higher priority during training. We evaluate our method through closed-loop experiments in the CARLA simulator. The results on Bench2Drive scenarios demonstrate the effectiveness of CAPS in enhancing model generalization, with substantial improvements in both driving score and success rate.**

***Index Terms*—Autonomous Driving, Trajectory Clustering, Priority Sampling, Learning-Based Planner, Closed-Loop Performance, CARLA Leaderboard 2.0, VQ-VAE**

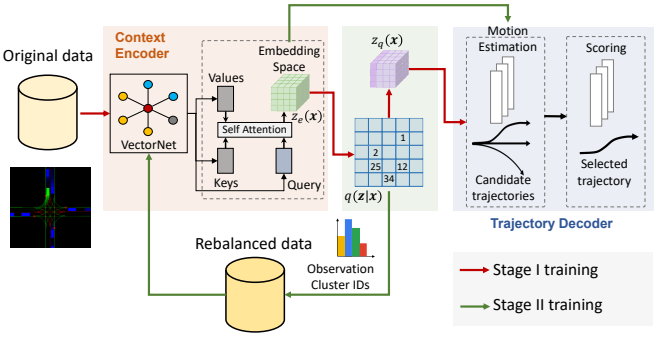


Fig. 1: Block diagram of the CAPS framework. In Stage I, a VQ-VAE module is introduced to facilitate the joint training of the clustering model and the planner. In Stage II, the trained VQ-VAE generates codebook IDs for the training samples, which are re-weighted based on cluster frequencies to refine the main planner.

I. Introduction

Imitation learning (IL) is a widely used approach for end-to-end training in autonomous driving (AD), where policies are learned from expert demonstrations. However, a significant challenge arises from the nature of expert datasets: they predominantly consist of trivial scenarios, including cruising in a straight line and decelerating at a stop sign, which can be easily handled even by a rule-based planner. Meanwhile, edge cases, such as parking cut-ins, sudden stops, and near-accident incidents, are rare and challenging to handle, and they might be overshadowed by the trivial scenarios as their frequencies are much lower.

The data imbalance leads to IL models overfitting to common driving behaviors while failing to generalize to rare but critical edge cases [1], [2]. Naively scaling up the dataset size, even though it is costly and time-consuming, is not efficient, as the trivial scenarios with minimal learning value are still dominating the dataset. As highlighted in [3], an agent trained on just 10% of the data can perform as well as one trained on the full dataset, underscoring that a substantial portion of uniformly collected data does not meaningfully contribute to learning robust driving policies. This data imbalance becomes even more problematic in closed-loop evaluation: while a planner trained on a uniformly sampled dataset may perform well in expectation, a single failure in an edge case can lead to catastrophic consequences [4], [5]. Thus, ensuring strong generalization to rare, high-risk situations is more critical than merely excelling in routine driving scenarios.

Data balancing is a potential solution to the data imbalance problem. However, this requires access to labels for the samples, which requires costly human labeling, which is not well-scalable to large datasets. Furthermore, the criteria given to the annotators to categorize the samples are subjective and depend on the application domain and the task at hand, and need to be changed with the changes of the objective. For example, the class labels that are used for the data balancing in the trajectory prediction task is different from the class labels that are used for the data balancing in the planning task, as in the trajectory prediction task the focus in on minimizing the prediction error in trajectory space, while in the planning task the focus is on maximizing the safety of the driving behavior even if it deviates from logged expert's demonstrations, e.g. having less human-like driving behavior.

Another possible solution is to use rule-based approaches, which do not require human labour for data annotation. For instance, we can apply k-nearest neighbors (KNN) clustering in the trajectory space to cluster the samples into different classes, and then use the cluster frequencies to re-weight the samples. However, this approach skips the context information, and it lacks the capacity to discriminate complex nuances in the dataset. For example, trajectory-based clustering

cannot differentiate between a scenario where the ego vehicle is decelerating due to reaching a red light and a scenario where the ego vehicle is decelerating as a result of observing a rare collision ahead. Similarly, in trajectory prediction, models that neglect causal inter-agent context have been shown to suffer from poor robustness and generalization [6].

To have a more effective scenario labeling to be used for data balancing, we propose CAPS (Context-Aware Priority Sampling), a novel framework that leverages variational self-supervised representation learning. CAPS adopts Vector Quantized Variational Autoencoders (VQ-VAE) [7] to conditionally encode the future trajectory of the ego-vehicle into a quantized latent representation given the scene context information. As a byproduct of the VQ-VAE trajectory encoding, the expert demonstrations are automatically clustered into several distinct classes. This approach is different from the traditional clustering approaches that focus solely on trajectory features, as CAPS captures rich contextual information that is effective in reconstructing the future trajectory of the ego-vehicle.

CAPS, similar to the recent generative models [8], [9], employs a two-stage training process, In stage I, first, the model is trained to encode the contextful future trajectory of the ego-vehicle into a quantized latent representation, and use the quantized codebook indices as the clustering labels for the data samples. We assign a sampling weight to each data sample based on the inverse of its cluster frequency. In stage II, which is the main training stage, we use a weighted sampling strategy to train the planner model with an enhanced focus on the underrepresented data samples. The training of the stages is decoupled, which allows us to focus on the quality of the representation learning and have a good clustering model regardless of the objective functions of the downstream task, here, the planning task. In stage II training, we only need the class-balanced weights from stage I, and the focus is on the performance of the planner model without missing the underrepresented scenario classes.

To summarize, we can highlight the following contributions of this work:

- We introduce CAPS (Context-Aware Priority Sampling), a novel framework designed to effectively learn a context-aware representation of the expert demonstrations, which is used for a class-balanced training for the planning task.
- We conduct extensive closed-loop experiments on the Bench2Drive benchmark dataset [10], demonstrating that our proposed CAPS framework consistently outperforms baseline methods. In particular, CAPS surpasses trajectory endpoint and anchor-based clustering approaches, which represent intuitive rule-based strategies for trajectory clustering. Furthermore, we show that CAPS achieves superior performance compared to other state-of-the-art methods with a similar computational budget in the planning task.

In Section III, we discuss the implementation details of CAPS. In Section IV, we demonstrate that CAPS achieves roughly a 10% improvement in Driving Score and Success Rate on Bench2Drive scenarios, illustrating the benefits of context-aware priority sampling for autonomous driving without the need for additional expert data or extra computational cost during deployment.

II. Related Works

Prior approaches on data balancing focus on modifying demonstration data before training through various preprocessing techniques, such as up-sampling minority clusters or down-sampling majority clusters. Many of these methods rely on SMOTE [11], which generates synthetic samples for the minority class by interpolating between minority samples and their k nearest neighbors in the latent space. For instance, studies in [12]–[14] explore up-sampling by creating synthetic instances for minority classes, while [15], [16] focus on down-sampling the majority class to balance the dataset. Another common strategy is data augmentation, which expands the training set through synthetic transformations [17]–[19]. For example, [18] perturbs demonstration trajectories by altering their midpoints with a uniform distribution, while keeping the start and end points intact. Although these techniques are sample-efficient and require fewer computational resources, they often depend on offline clustering or manual labeling to determine which samples should be perturbed or augmented—a process that becomes increasingly impractical and time-consuming as dataset size grows.

To mitigate dataset limitations during deployment, another group of methods leverages expert policies to improve closed-loop performance. The key idea is to continuously collect additional samples from both the expert and the trained policy, in contrast to behavior cloning [20], which relies on a fixed set of demonstrations. These methods dynamically update the dataset with states encountered by the learner during execution [1], but require collecting large numbers of expert-labeled samples, making them resource-intensive. Several works attempt to address these challenges [21]–[23]. For example, SafeDAgger [21] predicts unsafe trajectories and queries the expert only in those cases. While such approaches offer improvements in closed-loop performance, they still rely heavily on expert supervision, which limits their practicality. Moreover, they often depend on planner failures as a trigger for identifying valuable training data, which is inefficient and can result in suboptimal learning.

More recent work combines imitation learning with reinforcement learning to further enhance robustness. In these approaches, RL rewards guide models initially trained on demonstration data [24]–[27]. For example, [24] presents a hybrid method that leverages large-scale human driving data through IL and incorporates RL to handle rare and challenging scenarios. Similarly, [25] introduces a hierarchical model that blends IL and RL, improving generalization to unseen situations and navigation in complex environments. Despite their promise, these approaches typically require vast

amounts of data and computational resources, which can be impractical in real-world scenarios, particularly given the difficulty of achieving realistic autonomous driving simulations [28], [29].

Taken together, these methods highlight the need for solutions that can automatically identify and prioritize the most informative data samples for training and fine-tuning. Instead of relying on failures to signal valuable data, an effective framework should analyze the encoded context to proactively recognize infrequent yet critical cases. Prioritizing such samples during training can reduce manual effort, improve sample efficiency, and lead to more robust learning outcomes.

III. Methodology

A key challenge in training motion planning models is to determine which data samples most effectively contribute to improving closed-loop performance. One approach to address this is clustering data based on the main characteristics of the ego trajectory, such as position, speed, acceleration, curvature, etc. However, if this clustering is based solely on the ego vehicle's trajectory features, it may overlook crucial contextual information from the surrounding environment. Closed-loop performance is influenced not only by the ego trajectory but also by interactions with other agents and environmental factors. Ignoring this context can result in suboptimal outcomes during closed-loop tests. A holistic evaluation should therefore assess each sample by considering both the key features of the ego trajectory and the surrounding context. Unlike traditional VAEs that rely on continuous latent variables, VQ-VAE learns a finite set of latent codes, enabling a more structured and interpretable data representation [7]. By encoding contextual information, VQ-VAE ensures that similar inputs are mapped to nearby or identical latent codes, effectively capturing meaningful patterns and dependencies. This discrete representation is particularly beneficial as it reduces sensitivity to small variations, making clustering more robust and ensuring that learned representations remain stable across different inputs. Additionally, discrete latent spaces help mitigate posterior collapse, a common issue in VAEs where the latent space becomes less informative.

We employ a two-stage training process. In Stage 1, we train both VQ-VAE and our planner, where the planner operates in a generative mode to produce trajectories as outputs. Importantly, our planner is trained with the clustering model integrated into the training loop, ensuring that trajectory generation aligns with the learned discrete representations. Once both models are trained, Stage 2 deploys the clustering model to assign a cluster ID to each training sample. We then apply sample weighting based on cluster frequencies to balance the distribution of the training data for the planner, enhancing its ability to generalize across different scenarios.

A. Model Architecture

Figure 1 illustrates the architecture of our model, which integrates both the planner and the clustering modules. We describe each component in more detail as follows.

Context Encoder: The planner is composed of a context encoder and a trajectory decoder. For the context encoder, we employ VectorNet [30], which processes the observed and future states of the ego vehicle, surrounding objects, and map information. To simplify the notation, we use subscripts p and f to denote *past* and *future*, respectively. Specifically, $\mathbf{s}_{\mathrm{p}}^{\mathrm{ego}}$ and $\mathbf{s}_{\mathrm{f}}^{\mathrm{ego}}$ represent the past and future state sequences of the ego vehicle, while $\mathbf{s}_{\mathrm{p}}^{\mathrm{obj}}$ and $\mathbf{s}_{\mathrm{f}}^{\mathrm{obj}}$ correspond to the past and future states of surrounding objects. Additionally, $\mathbf{c}_{\mathrm{p}}$ and $\mathbf{c}_{\mathrm{f}}$ capture the past and future context of the scene. With these definitions, the input to the scene encoder is represented as:

$$\mathbf{x} = \left\{ (\mathbf{s}_{\mathrm{p}}^{\mathrm{ego}}, \mathbf{s}_{\mathrm{f}}^{\mathrm{ego}}), (\mathbf{s}_{\mathrm{p}}^{\mathrm{obj}}, \mathbf{s}_{\mathrm{f}}^{\mathrm{obj}}), (\mathbf{c}_{\mathrm{p}}, \mathbf{c}_{\mathrm{f}}) \right\}. \tag{1}$$

The embedded features are then passed to a transformer module to integrate information from all components of the scene. This allows the system to model the relationships between agents, road elements, and surrounding objects. We employ a multi-head attention module to enable the model to focus on multiple aspects of the scene simultaneously, learning diverse patterns of interaction that are essential for accurate decision-making in dynamic environments.

Trajectory Decoder: Our trajectory decoder module is designed based on [31], adapting its structure to better suit our planning framework. It consists of two key components: a trajectory generation module and a scoring module. The trajectory generation module produces a diverse set of candidate future trajectories, each conditioned on selected target points. These targets represent plausible endpoints within a predefined time horizon, ensuring that the generated trajectories align with feasible motion patterns in real-world driving scenarios. The scoring module evaluates and ranks these trajectories based on their plausibility, selecting the most suitable one for execution. To adapt this architecture for planning applications, we incorporate contingency masks, as introduced by [32]. These masks help mitigate compounding errors during sequential decision-making by filtering out unrealistic or unsafe trajectory candidates, leading to more reliable and stable planning in dynamic environments.

Clustering Module: Our clustering module is based on a VQ-VAE architecture, as shown in Fig. 1, enabling the discrete representation of continuous data. VQ-VAE quantizes the embedding space using a codebook, assigning each data sample to a cluster. Let $\mathbf{z}_e(\mathbf{x})$ denote the embedding representation of input $\mathbf{x}$, obtained from the context encoder module. The embedding space $\mathbf{z}_e(\mathbf{x})$ has dimensions $N_{\text{obj}} \times d_e$, where N_{obj} is the total number of objects in the scene, and d_e is the dimension of latent vectors. The first column of the embedding space corresponds to the ego vehicle's feature, encapsulating critical scene context from the ego's perspective. We denote this vector as $\mathbf{z}_e^{\mathrm{ego}}$ and use

it as input to our VQ module. The codebook consists of K embedding vectors $\{e_1, e_2, \ldots, e_K\}$, where each $e_j \in \mathbb{R}^{d_e}$ represents a discrete latent vector. Following the VQ-VAE pipeline [7], we quantize $\mathbf{z}_e^{\text{ego}}$ by finding the closest discrete latent variable e_k from the codebook:

$$k = \arg\min_j ||\mathbf{z}_e^{\text{ego}} - e_j||_2. \tag{2}$$

The quantized representation, which serves as input to the VQ decoder, is then given by $\mathbf{z}_e^{\text{ego}} = e_k$. The decoder output is used to reconstruct the input, ensuring meaningful latent representations. The losses used for training are introduced in the next section.

B. Model Training Approach

As previously described, our pipeline follows a two-stage training process. In the first stage, we jointly train the planner and the VQ-VAE model to determine the clustering labels for each data sample. During this phase, the planner functions as a generative model, ensuring that the model's input and output remain consistent while learning meaningful embedding representations. For the planner, we employ an imitation loss similar to [31], which aims to reconstruct the ego trajectory in the output. For the VQ model, we optimize the following loss function

$$\begin{aligned}\mathcal{L} = &\log p(\mathbf{z}_e^{\text{ego}}|\mathbf{z}_q^{\text{ego}}) + \\ &|sg([\mathbf{z}_e^{\text{ego}}] - e|^2 + \beta|\mathbf{z}_e^{\text{ego}} - sg[e]|^2.\end{aligned} \tag{3}$$

The first term represents the reconstruction loss, which ensures that the encoder and decoder accurately reconstruct the ego features. The second term, known as the codebook loss, minimizes the discrepancy between the encoder's output and the closest codebook vector. Here, $sg[.]$ denotes the stop-gradient operation, which prevents direct updates to the codebook while encouraging the encoder to align with the learned embeddings. Since the embedding space has no inherent scale, its magnitude can grow indefinitely if the embeddings e fail to update at the same rate as the encoder parameters. To address this, the third term, referred to as the commitment loss (introduced by [7]), ensures that the encoder consistently maps inputs to a specific embedding, preventing uncontrolled expansion.

In the second stage, we leverage the trained VQ-VAE model to assign a cluster label to each training sample. Based on cluster frequencies, we then adjust sample weights to balance the training data for the planner. The planner is subsequently trained using an imitation loss similar to the one proposed in [31], which aims to reconstruct the ego trajectory in the output. This approach improves the planner's ability to generalize across diverse scenarios by ensuring a more balanced representation of training samples across different clusters.

IV. Results and Discussions

A. Experiments Setup

Simulation Environment: We use CARLA Leaderboard 2.0 for all our experiments. CARLA Simulator [33] provides a highly detailed virtual environment to design and train AI models in tasks such as perception, decision-making, and control. CARLA Leaderboard 2.0 extends the original leaderboard by providing a standardized set of tasks and evaluation metrics for autonomous driving systems. It encompasses a diverse range of scenarios and challenges designed to assess multiple aspects of driving performance.

Data Collection: One major challenge in using CARLA Leaderboard 2.0 is that no single expert can consistently perform across its diverse scenarios. Inspired by [10], we developed an automated scenario generation module that segments each route in Leaderboard 2.0 into multiple short-duration segments, with each segment representing a distinct scenario. This approach allows us to compile a more balanced dataset for training our planner and efficiently filter out failing scenarios, compared to collecting data from full routes. Additionally, to further enhance our training data, we incorporate high-quality data provided by the official Leaderboard 2.0 team. Their CARLA logs feature manual execution of each scenario, all achieving a 100% score, and include pre-calculated vehicle trajectories for all elements within the scenario.

Evaluation Benchmark: We use Bench2Drive [10] for benchmarking. It provides a comprehensive set of 220 short-segment scenarios that capture various challenging traffic behaviors.

Evaluation Metrics: We focus on two key metrics reported in Bench2Drive:

- *Driving Score*: The driving score evaluates the quality of driving behavior by penalizing infractions such as collisions, off-road driving, and failure to adhere to traffic rules, as defined in the official CARLA Leaderboard 2.0.
- *Success Rate*: The success rate measures the percentage of scenarios in which the vehicle successfully reaches its goal without experiencing critical failures. A critical failure is defined as any infraction (with the exception of 'min-speed infraction') captured by the CARLA Simulator.

B. Experimental Results

We evaluate our approach on CARLA using both sensor inputs and privileged inputs. Privileged inputs are defined as a vectorized scene representation that is accessible by CARLA. In the experiment with the privileged inputs, the effect of the perception module is isolated, and the planner has direct access to a more compact and accurate representation of the environment. For sensor-based experiments, we deploy six cameras around the vehicle and fine-tune a version of the VAD backbone to estimate the vectorized scene representation [34], [35]. Our training dataset consists of approximately $385k$ samples collected using our rule-based planner agent, combined with around $178k$ samples obtained by replaying logs from the official CARLA Leaderboard 2.0 team. In our experiment, we set the number of codebook IDs to 64. As previously discussed, in the VQ-VAE implementation,

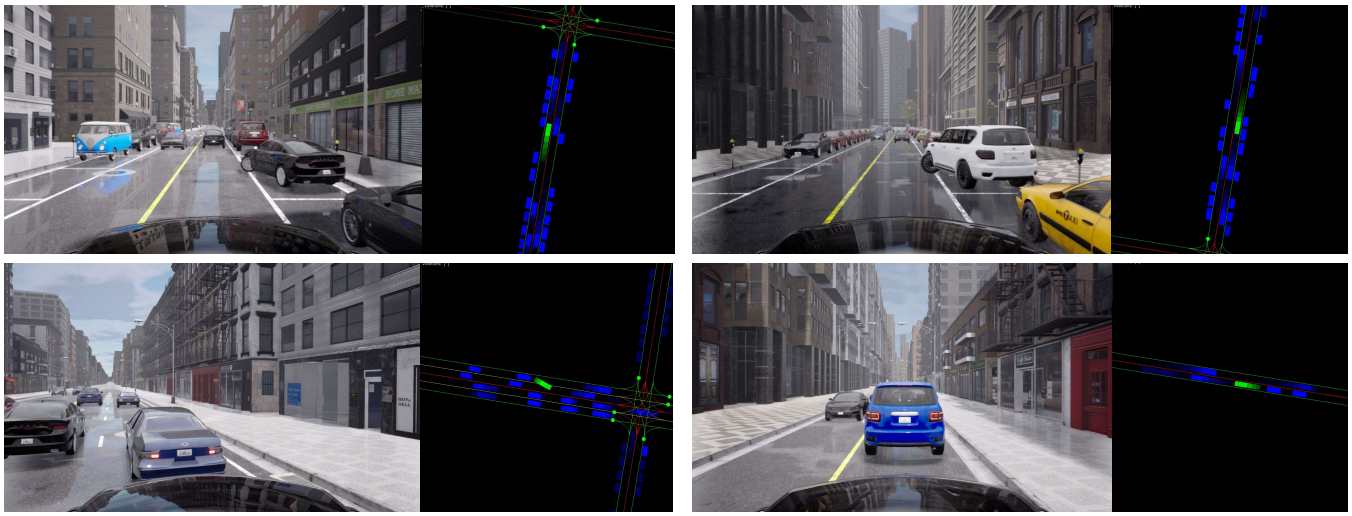


Fig. 2: Comparison of two examples of scenes clustered using our approach, where each row corresponds to the same codebook ID. The first row predominantly contains parking cut-in scenarios, while the second row features instances where the ego vehicle stops and waits behind a stationary vehicle or obstacle. In the bird's-eye-view images of the traffic scenes, the ego vehicle and the surrounding vehicles are shown in green and blue, respectively.

TABLE I: Planning performance for data balancing with different clustering methods. The results are presented for 220 short segment scenarios of Bench2Drive.

Input	Clustering Method	Driving Score↑	Success Rate(%)↑
Privileged	End-point	59.63	48.23
	Anchor-based	62.60	51.83
	PER	61.18	50.26
	CAPS	**68.91**	**56.97**
Sensor	End-point	57.06	46.16
	Anchor-based	60.64	48.69
	PER	60.42	47.98
	CAPS	**66.76**	**52.87**

TABLE II: Effect of contextual information during clustering. Removing the surrounding context increases the time required to complete scenarios despite similar infraction statistics.

Model	Avg. Comp. Time (s)↓	Route Comp.(%)↑	Success Rate(%)↑	Driving Score↑
No Clustering	71.4	92.3	48.2	59.5
No-Agent Context	69.3	90.1	51.8	66.6
No-Agent/Map Context	74.4	89.8	49.7	63.8
CAPS	**50.0**	**97.3**	**52.9**	**66.7**

the codebook IDs are assigned to the quantized embedding vectors in the latent space, which also serves as the cluster ID for data samples used in Stage I of our pipeline.

Table I reports the closed-loop performance of our planner across different clustering and rebalancing strategies on 220 short-segment scenarios from Bench2Drive [10], considering both privileged and sensor inputs. We evaluate two rule-based clustering strategies: endpoint clustering, where the ego trajectory is represented by its start and end points, and anchor-based clustering, which uses the full 3-second ego trajectory. Both methods employ k-means to partition the dataset into 64 clusters, providing a consistent comparison baseline. In contrast, CAPS encodes both the ego trajectory and its surrounding context in the scene embeddings while maintaining the same number of clusters. Results show that CAPS consistently outperforms the rule-based approaches, emphasizing the value of context-aware representations for planner performance. We also investigate Prioritized Experience Replay (PER) sampling [36], where training samples are weighted by their loss values, and higher-loss samples are sampled more frequently. We find that this strategy does not significantly improve imitation learning performance, as planner training should prioritize closed-loop metrics such as success rate and driving score. Adjusting sample weights based on open-loop metrics alone provides no measurable benefit, as confirmed by our evaluation results.

To further analyze the importance of contextual information in the clustering stage, we perform additional context-exclusion ablations during Stage I training. The results of these studies are reported in Table II, including: i) **No-Agent Context**, where we excluded the all of the agents from the scene, so clustering is done using ego trajectory and the map elements, ii) **No-Agent/Map Context**, where we excluded both agents and the map elements from the scene. Regarding the evaluation metrics, we report average scenario completion time, route completion percentage, success rate, and driving score. According to the results, CAPS outperforms the ablated models and the baseline in all metrics. While the gain in the driving score metric is relatively small, we observe that there is a larger improvement in the other metrics; the scenario completion time for CAPS is 32% lower than No-Agent/Map, and it is 28% lower than the No-Agent model.

Similarly, for the route completion metric, CAPS gets major improvements. These results indicate that incorporating scene context during clustering leads to improved performance of the learned planner.

To provide additional evidence of the effectiveness of our approach in clustering scenes, we present different samples with the same cluster IDs in Fig. 2. Each row in this figure corresponds to a distinct cluster ID, highlighting the consistency of the cluster characteristics across different scenes. The first row presents scene samples corresponding to the parking cut-in scenario, while the second row shows scenarios where the ego vehicle stops and waits behind a stationary vehicle or obstacle. Despite coming from different scenes, the nature of the scenarios within the same cluster is similar.

To examine temporal consistency and continuity across consecutive frames, we select a segment of a driving scenario where the ego vehicle travels along a two-way, double-lane road until it encounters lane congestion caused by an accident (Fig. 3). Because the scene contains multiple dynamic agents, we highlight only the most distinguishable key frames—those where a human expert can intuitively map the evolving situation to the transitions that our method identifies as critical. By probing the temporal changes of the embedding space and retrieving the camera frames associated with the large changes, we observe three key moments:

1) when the ego vehicle first perceives the congestion, decelerates sharply to avoid collision, and begins negotiating with vehicles in the adjacent lane to bypass the blockage (Fig. 3)-A);
2) when a sufficient gap opens in the left lane, allowing the ego vehicle to change lanes safely (Fig. 3)-B);
3) when the ego vehicle clears the accident scene and attempts to merge back into the right lane (Fig. 3)-C).

Ideally, the codebook IDs and their corresponding embeddings before and after these transitions should capture the changes in the scene. Fig. 3) illustrates this by showing the temporal differences in the embedding space produced by the VQ-VAE module. Each sudden jump in embedding distance reflects a key transition. We also note sporadic jumps in the temporal embedding distances, resulting from CARLA abruptly spawning vehicles, which induce significant scene and cluster ID changes.

Table III compares the closed-loop performance of our planner trained using IL with other advanced end-to-end planners. We also include a comparison with an expert classical planner—a rule-based planner manually fine-tuned using privileged perception data from CARLA [35], [42]. While the planner trained without CAPS demonstrates reasonable performance, its performance does not surpass the baselines. By incorporating CAPS into the training pipeline and rebalancing the dataset, however, the same planner consistently outperforms all baselines. It highlights the effectiveness of our approach in prioritizing the most valuable samples. It is worth noting that some recent approaches, such as SimLingo (CarLLaVa) [43] could achieve higher performance than our base planner models. However, these models operate in a different scope and rely on much larger vision-language models (VLMs), with up to 100× more parameters, which require significantly higher training/inference resources. Therefore, we excluded VLM-based approaches to ensure a fair comparison.

TABLE III: Planning performance benchmarking with closed-loop simulation in CARLA. The results are presented for 220 short segment scenarios of Bench2Drive. Comparison is done with other approaches using similar computational budgets.

Input	Method	Driving Score↑	Success Rate(%)↑
Privileged	IL with Uniform Sampling	62.26	54.16
	CAPS (Ours)	68.91	56.97
	Expert	82.37	84.29
Sensor	AD-MLP [37]	18.05	00.00
	UniAD-Base [38]	45.81	16.36
	VAD [34]	42.35	15.00
	TCP-traj [39]	59.90	30.00
	ThinkTwice [40]	62.44	31.23
	DriveAdapter [41]	64.22	33.08
	IL with Uniform Sampling	59.49	48.16
	CAPS (Ours)	**66.76**	**52.87**
	Expert	75.82	82.72

V. Conclusions

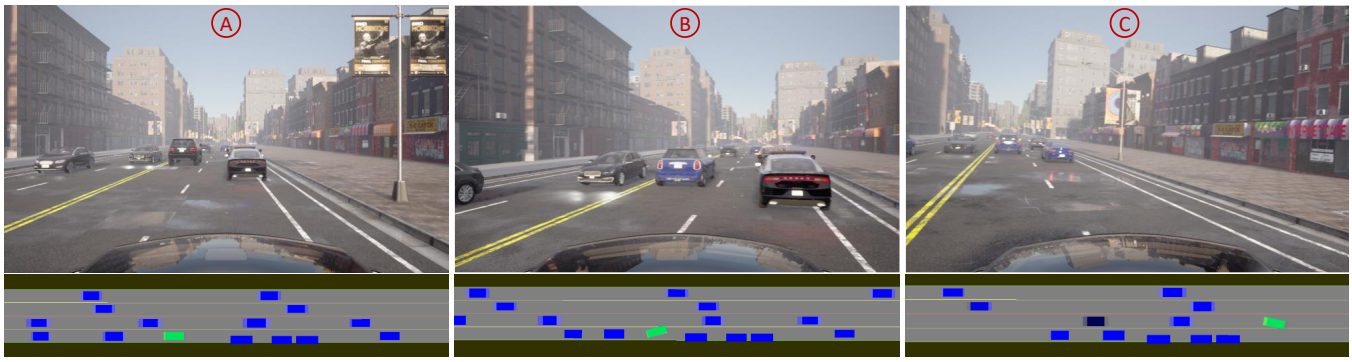
In this study, we introduced a context-aware priority sampling framework and demonstrated its effectiveness in enhancing the performance of a learning-based motion planner. By prioritizing rare and challenging samples that may otherwise be underrepresented during training, the framework improves sample efficiency and closed-loop robustness. Beyond planner training, our approach can also be applied during the data-collection stage to selectively capture high-value driving experiences, thereby reducing the storage and processing of redundant or low-information data. This is particularly important given that a single autonomous vehicle can generate terabytes of raw data annually, making real-time identification of the most informative samples critical for scalable fleet-wide training [44], [45]. Future research directions include investigating alternative VQ-VAE architectures such as [46], integrating the framework with closed-loop training pipelines as in [32], and analyzing their joint impact on trajectory clustering and priority sampling effectiveness.

References

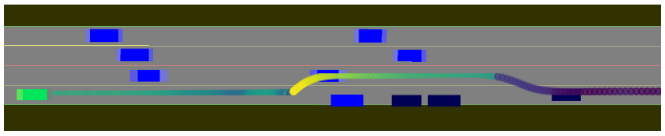
[1] S. Ross, G. Gordon, and D. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-regret Online Learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[2] S. Ross and D. Bagnell, "Efficient Reductions for Imitation Learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

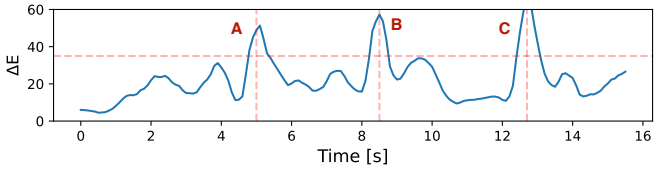
[3] E. Bronstein, S. Srinivasan, S. Paul, A. Sinha, M. O'Kelly, P. Nikdel, and S. Whiteson, "Embedding synthetic off-policy experience for autonomous driving via zero-shot curricula," in *Conference on Robot Learning (CoRL)*, 2022.



(a)



(b)



(c)

Fig. 3: Qualitative temporal probing of a driving scenario with CAPS. (a) Key transitions in a driving scenario are captured by the changes in the CAPS's quantized embedding space. The ego vehicle encounters lane congestion due to an accident and undergoes three main transitions: (A) slowing down and negotiating with adjacent vehicles, (B) changing lanes to bypass the blockage, and (C) returning to the right lane after passing the accident. (b) The ego trajectory is colored according to its cluster ID. To obtain these colors, the scene embeddings are first reduced using t-SNE, and then each point is assigned a color based on its cluster. (c) Temporal embedding distance of the scene. Sudden jumps in embedding distance (vertical dashed lines in this figure) correspond to these key frames.

[4] P. de Haan, D. Jayaraman, and S. Levine, "Causal confusion in imitation learning," in *Neural Discrete Representation Learning*, 2019.

[5] H. Liu, S. Dass, R. Martín-Martín, and Y. Zhu, "Model-based runtime monitoring with interactive imitation learning," in *International Conference on Robotics and Automation (ICRA)*, 2024.

[6] E. Ahmadi, S. Alizadeh, R. Mercurius, and A. Rasouli, "Curb your attention: Causal attention gating for robust trajectory prediction in autonomous driving," in *International Conference on Robotics and Automation (ICRA)*, 2025.

[7] A. van den Oord, O. Vinyals, and k. kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing System (NeurIPS)*, 2017.

[8] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[9] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International Conference on Machine Learning (ICML)*, 2021, pp. 8821–8831.

[10] X. Jia, Z. Yang, Q. Li, Z. Zhang, and J. Yan, "Bench2Drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[12] M. Temraz and M. T. Keane, "Solving the class imbalance problem using a counterfactual method for data augmentation," *Machine Learning with Applications*, vol. 9, p. 100375, 2022.

[13] M. Koziarski, M. Woźniak, and B. Krawczyk, "Combined cleaning and resampling algorithm for multi-class imbalanced data with label noise," *Knowledge-Based Systems*, vol. 204, 2020.

[14] K. Lei, Y. Xie, S. Zhong, J. Dai, M. Yang, and Y. Shen, "Generative adversarial fusion network for class imbalance credit scoring," *Neural Computing and Applications*, vol. 32, no. 12, 2020.

[15] K. Niu, Z. Zhang, Y. Liu, and R. Li, "Resampling ensemble model based on data distribution for imbalanced credit risk evaluation in p2p lending," *Information Sciences*, vol. 536, pp. 120–134, 2020.

[16] A. Guzmán-Ponce, J. Sánchez, R. Valdovinos, and J. Marcial-Romero, "Dbig-us: A two-stage under-sampling algorithm to face the class imbalance problem," *Expert Systems with Applications*, vol. 168, p. 114301, 2021.

[17] H. Mirkhani, B. Khamidehi, and K. Rezaee, "Augmenting safety-critical driving scenarios while preserving similarity to expert trajectories," in *Intelligent Vehicles Symposium (IV)*, 2024.

[18] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst," in *Robotics: Science and Systems (RSS)*, 2019.

[19] M. Ning, E. Sangineto, A. Porrello, S. Calderara, and R. Cucchiara, "Input perturbation reduces exposure bias in diffusion models," *arXiv preprint arXiv:2301.11706*, 2023.

[20] A. O. Ly and M. Akhloufi, "Learning to drive by imitation: An overview of deep behavior cloning methods," *Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 195–209, 2021.

[21] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," 2016.

[22] Y. Bicer, A. Alizadeh, N. K. Ure, A. Erdogan, and O. Kizilirmak, "Sample efficient interactive end-to-end deep learning for self-driving cars with selective multi-class safe dataset aggregation," in *International Conference on Intelligent Robots and Systems (IROS)*, vol. abs 1604 7316, 2019.

[23] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "Ensembledagger: A bayesian approach to safe imitation learning," 2019.

[24] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, R. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson *et al.*, "Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios," in *International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[25] E. Bronstein, M. Palatucci, D. Notz, B. White, A. Kuefler, Y. Lu,

S. Paul, P. Nikdel, P. Mougin, H. Chen *et al.*, "Hierarchical model-based imitation learning for planning in autonomous driving," in *International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[26] J. Booher, K. Rohanimanesh, J. Xu, and A. Petiushko, "CIMRL: Combining imitation and reinforcement learning for safe autonomous driving," *arXiv preprint arXiv:2406.08878*, 2024.

[27] X. Liu, T. Yoneda, R. L. Stevens, M. R. Walter, and Y. Chen, "Blending imitation and reinforcement learning for robust policy improvement," *arXiv preprint arXiv:2310.01737*, 2023.

[28] M. Alban, E. Ahmadi, R. Goebel, and A. Rasouli, "Getting SMARTER for motion planning in autonomous driving systems," in *Intelligent Vehicles Symposium (IV)*, 2025.

[29] E. Ahmadi and H. Schofield, "RLFTSim: Multi-agent traffic simulation via reinforcement learning fine-tuning (technical report for waymo open sim agents challenge)," Tech. Rep., 2025, technical Report. [Online]. Available: https://rlftsim.github.io/

[30] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "VectorNet: Encoding hd maps and agent dynamics from vectorized representation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[31] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, C. Li, and D. Anguelov, "TNT: Target-driven Trajectory Prediction," in *Conference on Robot Learning (CoRL)*, 2021.

[32] F. Arasteh, M. Elmahgiubi, B. Khamidehi, H. Mirkhani, W. Zhang, C. Tongtong, and K. Rezaee, "Validity learning on failures: Mitigating the distribution shift in autonomous vehicle planning," 2024.

[33] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," 2017.

[34] B. Jiang, S. Chen, Q. Xu, B. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang, "Vad: Vectorized scene representation for efficient autonomous driving," *arXiv preprint arXiv:2303.12077*, 2023.

[35] W. Zhang, M. Elmahgiubi, K. Rezaee, B. Khamidehi, H. Mirkhani, F. Arasteh, C. Li, M. A. Kaleem, E. R. Corral-Soto, D. Sharma, and T. Cao, "Analysis of a modular autonomous driving architecture: The top submission to carla leaderboard 2.0 challenge," 2024.

[36] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[37] J.-T. Zhai, Z. Feng, J. Du, Y. Mao, J.-J. Liu, Z. Tan, Y. Zhang, X. Ye, and J. Wang, "Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes," *arXiv preprint arXiv:2305.10430*, 2023.

[38] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, and e. a. Wang, Wenhai, "Planning-oriented autonomous driving," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 17 853–17 862.

[39] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[40] X. Jia, P. Wu, L. Chen, J. Xie, C. He, J. Yan, and H. Li, "Think twice before driving: Towards scalable decoders for end-to-end autonomous driving," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[41] X. Jia, Y. Gao, L. Chen, J. Yan, P. L. Liu, and H. Li, "Driveadapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving," in *International Conference on Computer Vision (ICCV)*, 2023.

[42] W. Zhang, P. Yadmellat, and Z. Gao, "Spatial optimization in spatio-temporal motion planning," in *Intelligent Vehicles Symposium (IV)*, 2022, pp. 1248–1254.

[43] K. Renz, L. Chen, E. Arani, and O. Sinavski, "Simlingo: Vision-only closed-loop autonomous driving with language-action alignment," in *Computer Vision and Pattern Recognition (CVPR)*, 2025.

[44] H. Lu, X. Jia, Y. Xie, W. Liao, X. Yang, and J. Yan, "ActiveAD: Planning-oriented active learning for end-to-end autonomous driving," 2024.

[45] R. Greer, B. Antoniussen, M. V. Andersen, A. Møgelmose, and M. M. Trivedi, "The why, when, and how to use active learning in large-data-driven 3d object detection for safe autonomous driving: An empirical exploration," 2024.

[46] F. Mentzer, D. Minnen, E. Agustsson, and M. Tschannen, "Finite scalar quantization: VQ-VAE made simple," in *International Conference on Learning Representations (ICLR)*, 2024.