ENHANCING REINFORCEMENT LEARNING FOR AUTONOMOUS DRIVING WITH OUTCOME-RELATED REWARD SHAPING

Anonymous authors
Paper under double-blind review

ABSTRACT

Designing effective reward functions for autonomous driving remains a major challenge since high-level human driving concepts are difficult to translate into precise numerical signals. While current approaches attempt to address this using expert-designed rewards or semantic signals from language models, these approximations often fail to align with the actual physical outcomes of the driving task. To address this, we propose a reward shaping method utilizing empirical environment outcomes as the evaluation target. Our approach introduces Outcome-Derived State Evaluation (ODSE), a state evaluator learned from reference trajectories by using trajectory-level terminal outcomes as supervised targets for visited states. By integrating these evaluations through the potential-difference formulation of Potential-Based Reward Shaping (PBRS), we derive a dense shaping signal that improves sample efficiency while only adding a controlled, explicitly characterized outcome term to the learning objective. Evaluations on the VLM-RL benchmark show that our method outperforms state-of-the-art reward models, thereby demonstrating that outcome-related signals can successfully enhance training effectiveness and establish a more robust driving strategy across diverse scenarios.

1 INTRODUCTION

In recent years, the field of autonomous driving has experienced rapid growth. However, finding driving policies that are both secure and capable of generalizing across various traffic scenarios still remains a major challenge (Cao et al., 2022; Feng et al., 2023). Reinforcement Learning (RL) has emerged as a powerful framework for this task, enabling agents to learn end-to-end control by optimizing for long-term performance through environment interaction.

The design of the reward function is a critical component of RL learning process, as it serves as the primary signal guiding the agent's behavior. However, formulating an effective reward for autonomous driving is a difficult task, since the criteria for 'good driving' are often semantically ambiguous and difficult to formally encode into a numerical reward function. Currently, these efforts largely fall into categories such as expert-designed rewards and model-generated rewards (Nie et al., 2025). Traditional expert-designed approaches (Cao et al., 2022; Dosovitskiy et al., 2017; Zhang et al., 2024), such as Chen et al. (Chen et al., 2021), rely on a carefully balanced set of manually designed penalties for unsafe actions and rewards for desirable behaviors. More recent methods automate the reward engineering process by leveraging the reasoning capabilities of Large Language Models (LLMs) to synthesize or refine reward functions from high-level task descriptions (Han et al., 2024; Hazra et al., 2024). Similarly, Vision-Language Model-based (VLM) approaches (Baumli et al., 2023; Rocamonde et al., 2023), such as LORD (Ye et al., 2025) or VLM-RL (Huang et al., 2025), attempt to derive reward signals by mapping visual observations to semantic goals.

Despite these advancements, existing methods primarily focus on whether an immediate state appears 'good' based on external semantic approximations (e.g., speed, center deviation), without explicitly evaluating its long-term viability for the final outcome. This creates a gap where a reward model's linguistic objectives may not fully capture the relationship between the agent's states and the actual physical outcomes. To bridge this gap, we propose a grounded approach that derives evaluation signals directly from the outcomes of an agent's experience. Specifically, we leverage a reference agent to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

collect state trajectories and their eventual episode outcomes, and learn the empirical relationship between intermediate states and these outcomes. By assigning trajectory-level terminal outcomes as supervised targets to visited states, we transform sparse outcome information into a state value evaluator that can provide dense, step-wise guidance to a learning agent.

In this paper, we propose a reward shaping method with outcome-related evaluations to improve the training of driving agents. Unlike semantic reward models that interpret "good driving" through high-level descriptions, our method uses a learned memory module to analyze the state trajectories and terminal scores of a reference agent. From this analysis, we derive our proposed Outcome-Derived State Evaluation (ODSE), a state evaluator learned from the historical outcomes of reference trajectories. To integrate these signals without directly accumulating absolute ODSE evaluations throughout a trajectory, we employ the potential-difference formulation of Potential-Based Reward Shaping (PBRS) (Ng et al., 1999), with ODSE acting as the potential function. Through this combination, we deliver dense step-wise guidance while only adding a controlled, explicitly characterized outcome term to the learning objective. We evaluate our framework using the VLM-RL benchmark (Huang et al., 2025), comparing it against four state-of-the-art baselines including both expert-designed (Zhang et al., 2024) and model-based (Hazra et al., 2024; Ye et al., 2025; Huang et al., 2025) reward functions. Our experimental results demonstrate that our proposed method establishes a superior and more robust driving strategy. Furthermore, our ablation studies empirically reveal that our shaping signals successfully extract and evaluate distinct driving patterns from various reference agents to effectively guide policy training.

In summary, this paper makes three contributions. First, we propose a novel reward shaping approach designed to bridge the "semantic gap" in autonomous driving by grounding evaluation signals in empirical outcomes rather than external linguistic approximations. Second, we introduce ODSE, a simple and grounded state evaluator learned from trajectory-level outcome supervision, and integrate it through the potential-difference formulation of PBRS to improve sample efficiency while keeping the change to the learning objective controlled and explicit. Third, we validate our approach on the VLM-RL benchmark against state-of-the-art expert-designed and model-generated reward functions, thereby demonstrating that outcome-related reward shaping can successfully enhance training effectiveness and establish a more robust driving strategy across diverse scenarios.

2 Related Work

Evaluating states and actions with effective reward signals is a pivotal challenge in reinforcement learning (RL), particularly when training agents for complex, real-world autonomous driving environments. Overcoming this requires robust reward engineering, which includes both the foundational design of the reward function and supplementary reward shaping. Currently, reward designs for autonomous driving tasks fall into three primary categories: expert-designed, LLM-based, and VLM-based reward structures.

Traditional reward engineering relies on manually crafted functions (Dosovitskiy et al., 2017; Cao et al., 2022; Chen et al., 2021; Zhang et al., 2024) to balance objectives from human experts. For instance, Chen et al. designed a reward function that penalizes collisions and lane deviations while incentivizing forward velocity and steering stability (Chen et al., 2021). ChatScene (Zhang et al., 2024) utilizes structured weighted sums of reward terms which encourage smooth driving behaviors (longitudinal speed, lateral acceleration, and steering control) while penalizing unsafe actions (collisions, out-of-lane driving, speeding). However, these expert-designed signals require significant domain expertise and can lead to sub-optimal policies if the reward components are poorly balanced.

To overcome the limitations of manual design, several recent frameworks (Han et al., 2024; Hazra et al., 2024) leverage Large Language Models (LLMs) to enhance expert-designed rewards with simple rules. AutoReward (Han et al., 2024) integrates Chain-of-Thought environment code analysis and task requirement decomposition to refine their reward function. Focusing exclusively on the foundational reward, REvolve (Hazra et al., 2024) frames reward design as an evolutionary search problem, leveraging LLMs and human feedback to iteratively create more comprehensive, human-aligned reward functions.

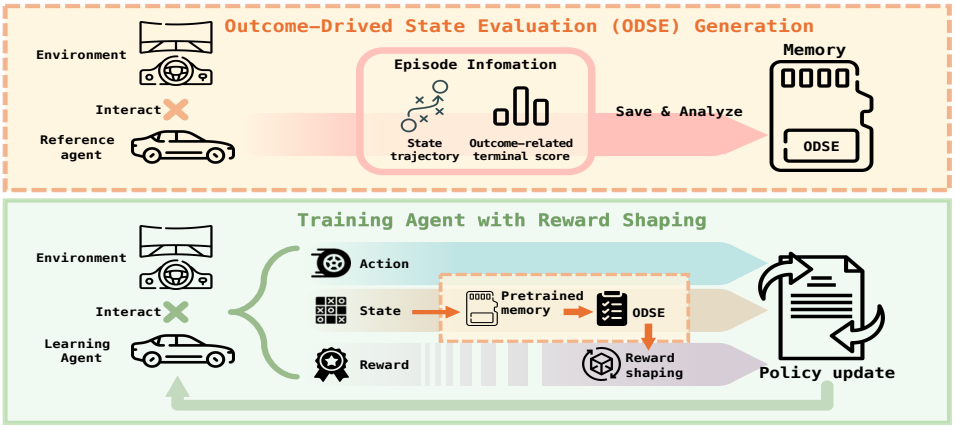


Figure 1: The overview of the driving agent training with our ODSE-based reward shaping. A detailed description of this figure is provided in Section 3.2.

Another prominent trend involves using Vision-Language Models (VLMs) (Rocamonde et al., 2023; Sontakke et al., 2023; Ye et al., 2025; Huang et al., 2025; Fu et al., 2025; Wasif et al., 2025) to derive reward signals from both visual and textual inputs. VLM-RM (Rocamonde et al., 2023) shows that VLMs can be trained as zero-shot reward models (RM) based on linguistic goals. LORD (Ye et al., 2025) focuses on generating negative reward shaping signals through VLMs to penalize unsafe behaviors, while VLM-RL (Huang et al., 2025) utilizes a Contrasting Language Goal (CLG)-as-reward paradigm, using both positive and negative language goals to generate semantic reward shaping terms.

3 Methodology

3.1 Autonomous Driving Agent

The training environment for the autonomous driving agent is modeled as a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ represents the state transition function, where $\Delta(\mathcal{S})$ denotes the set of probability distributions over $\mathcal{S}$. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ maps transitions to local rewards based on the current state s_t, the action a_t, and the subsequent state s_{t+1}. The agent's objective is to learn an optimal policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted return: $G(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t R_t\right]$, where $\gamma \in [0, 1]$ is the discount factor and T represents the terminal step of an episode.

3.2 Overview

In this paper, we propose a reward shaping method that leverages Outcome-Derived State Evaluation (ODSE) to enhance the learning of end-to-end autonomous driving policies. As shown in Figure 1, our training framework is separated into two distinct stages: the generation of ODSE and the subsequent training of the driving agent with reward shaping. During the generation stage, a reference driving agent is implemented to interact with the environment to collect extensive episode information, specifically capturing state trajectories and outcome-related terminal scores. This data is then analyzed and stored in a specialized Memory module, which generates the ODSE by correlating historical state sequences with their eventual outcomes.

In the training stage, this pre-trained memory is integrated into the learning loop to provide step-wise guidance. As the learning agent interacts with the environment, the memory module provides the corresponding ODSE for each visited state, which the reward shaping module then uses to augment the local environmental rewards. In the following, we mainly elaborate on the details of the ODSE generation and ODSE-based reward shaping procedure.

3.3 Outcome-Derived State Evaluation Generation

Previous end-to-end autonomous driving policy learning methods primarily rely on expert-designed or language model-based rewards. While these functions evaluate states based on specific physical constraints (e.g., vehicle distance, speed, center deviation), they do not contain direct information regarding whether a given state will lead to successful goal fulfillment in future steps. The most intuitive way to address this is through a sparse reward simply rewarding route completion and penalizing failures. However, this approach has proven ineffective for training drivable autonomous systems (Cao et al., 2022), as the feedback is far too infrequent to guide continuous, complex maneuvers. Instead, our key insight is to model the statistical relationship between intermediate states and global episode outcomes.

As illustrated in Figure 1, we utilize a memory to collect and store episode trajectories. Let an episode τ consist of a state trajectory $\tau = \{s_1, s_2, \ldots, s_T\}$, where s_T denotes the terminal state. Upon termination, an outcome-related terminal score $sc(\tau)$ is assigned to characterize the episode's performance (e.g., success or failure in reaching a predefined goal). To evaluate the states within τ, we assign the same $sc(\tau)$ to every single state in the episode τ and save the states in the memory. In this way, the same terminal signal that is too sparse to serve as the sole online reward (appearing once per episode) becomes a dense supervised label (one regression label per state). This regression target is well-defined regardless of how frequent or rare success is in the reference data, since both successes and failures provide supervision. However, the reliability of the learned evaluation for a learner state depends on how well the reference-state distribution covers similar states. We empirically examine sensitivity to different reference policies in Section 4.3.3. Over multiple episodes, this memory effectively forms a dataset $\mathcal{D}$ of collected trajectories. Within this memory, the quality of a specific state s can be quantified as the average terminal score of all episodes that included s. We define this metric as the **Outcome-Derived State Evaluation (ODSE)**:

$$\tilde{V}(s) = \frac{\sum_{\tau_i \in \mathcal{D}} \mathbb{I}(s \in \tau_i) \cdot sc(\tau_i)}{\sum_{\tau_i \in \mathcal{D}} \mathbb{I}(s \in \tau_i)}, \tag{1}$$

where $\tilde{V}(s)$ represents the ODSE for state s, $sc(\tau_i)$ is the terminal score of episode τ_i and $\mathbb{I}(s \in \tau_i)$ is an indicator that equals 1 if state s is visited in episode τ_i, and 0 otherwise. Formally, the ODSE is a Monte Carlo (MC) estimate of the reference policy's expected outcome under a non-discounted ($\gamma = 1$) sparse-terminal-reward setting. It is learned from the reference policy's own data and is later used off-policy with respect to the learning agent.

Technically, we implement the ODSE generation with a neural network-based memory. This architecture consists of a Transformer encoder (Vaswani et al., 2017) followed by a multi-layer perceptron (MLP). Once an episode ends, the states within the episode are encoded into latent representations by the Transformer, which are subsequently mapped to the episode terminal score by the MLP. We adopt this specific design for two primary reasons. First, the Transformer encoder can efficiently process the high dimensionality of states inherent in autonomous driving environment. Second, it bridges a critical gap between theory and practice: while ODSE theoretically requires averaging over identical states, the vastness of the state space makes some state revisitings highly improbable. By learning to generalize across similar latent representations, the network overcomes this limitation by functioning as a learned k-Nearest Neighbors aggregation process. The pseudo-code of memory training and the implementation details are shown in Appendix A.1.

3.4 Reward Shaping with Outcome-derived State Evaluation

Leveraging the ODSE, we incorporate these empirical evaluations into the learning objective. Although directly adding ODSE values to the reward appears straightforward, such an approach can introduce training instability. To address this, we adopt the potential-difference structure of Potential-Based Reward Shaping (PBRS) (Ng et al., 1999) to control the incorporation of ODSE by avoiding the accumulation of absolute state evaluations throughout a trajectory. Formally, the potential-based shaping term $\delta(s_t)$ is calculated as:

$$\delta(s_t) = \begin{cases} \gamma \tilde{V}(s_{t+1}) - \tilde{V}(s_t) & \text{if } s_t \text{ is non-terminal} \\ 0 & \text{if } s_t \text{ is terminal} \end{cases}, \tag{2}$$

where $\tilde{V}$ is the ODSE function and γ is the discount factor defined in Section 3.1. For brevity, we refer to this formulation as **PBRS** hereafter. The revised reward r'_t is subsequently formulated as:

$$r'_t = r_t + \epsilon \cdot \delta(s_t), \tag{3}$$

where r_t is the local reward, and ϵ is the coefficient of the reward shaping.

For an episode, the cumulative impact of the shaping on the discounted return is the difference $\epsilon \cdot (\gamma^{T-t}\tilde{V}(s_T) - \tilde{V}(s_t))$, which contains two parts with different roles. The state part $\tilde{V}(s_t)$ only depends on the current state, so it works as a value baseline (Williams, 1992) that does not change the expected policy gradient. The terminal part $\gamma^{T-t}\tilde{V}(s_T)$ adds the predicted terminal outcome to the learning objective with a small weight ϵ. Therefore, the learner optimizes the local reward in a controlled way, and $\epsilon \to 0$ recovers the original objective. The full analysis is shown in Appendix A.2. Furthermore, since the ODSE memory is pre-trained and frozen, our shaped environment remains a stationary MDP, avoiding the nonstationarity that may arise from dynamic shaping signals. The pseudo-code is also shared in Algorithm 2 in Appendix A.1.

4 Evaluation

4.1 Main Experiment setup

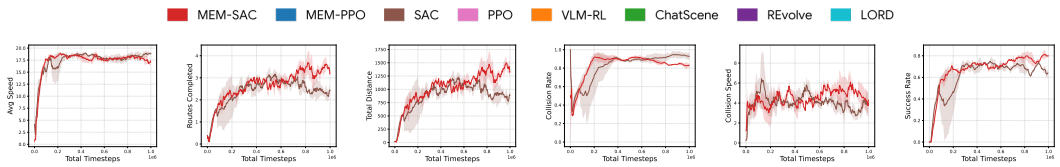
To evaluate the effectiveness of our proposed framework, we conduct our experiments using the VLM-RL benchmark (Huang et al., 2025), which is built upon the Stable-Baselines3 library (Raffin et al., 2021) and the CARLA 0.9.13 simulator (Dosovitskiy et al., 2017). This benchmark supports end-to-end driving policy training and integrates various state-of-the-art reward design methods, ranging from expert-defined structures (Chen et al., 2021; Cao et al., 2022; Wang et al., 2023; Zhang et al., 2024) to rewards generated by LLMs or VLMs (Baumli et al., 2023; Hazra et al., 2024; Rocamonde et al., 2023; Ye et al., 2025). The environment from the benchmark involves five distinct town maps populated with 20 CARLA autopilot-controlled vehicles to simulate realistic traffic. Navigation routes are dynamically generated with randomized start and end points. An episode is terminated if the training agent collides with an object, remains stationary for more than 90 seconds, or deviates from the road centerline by more than 3 meters. A **successful** episode is defined by the agent reaching a cumulative driving distance of 3,000 meters, which may contain multiple individual routes.

We design our main experiments to address two questions: (Q1) Does adding ODSE to a local reward significantly enhance training effectiveness? and (Q2) How does our ODSE-enhanced reward compare to state-of-the-art reward design methods?

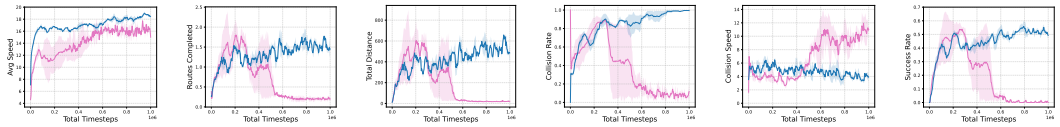
For Q1, we utilize PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018) as base learners, both using the default benchmark local reward. We pre-train vanilla PPO and SAC agents and then collect 5×10^5 steps of trajectories from each agent, which are used to train two distinct ODSE memory modules. For ODSE training, each reference route is assigned a sparse terminal outcome score of $+1$ for success and -1 for failure and these scores are used only as supervision. The resulting memory modules provide shaping signals to their respective learners, denoted as MEM-PPO and MEM-SAC, using a shaping coefficient of $\epsilon = 0.5$. Following standard VLM-RL protocols, agents are trained in Town 2 for 1×10^6 steps across three seeds. We also provide the parameter analysis for the memory training steps and shaping coefficient in Appendix A.3, and the cross-town training and evaluation in Appendix A.4.

For Q2, we compare our approach against four representative baselines from the VLM-RL benchmark: expert-designed ChatScene (Zhang et al., 2024), LLM-based REvolve (Hazra et al., 2024), VLM-based LORD (Ye et al., 2025), and VLM-RL (Huang et al., 2025). While ChatScene and REvolve utilize their own generated local rewards, VLM-RL and LORD share a common local reward and apply their respective proposed shaping terms. This is also the benchmark local reward used for the PPO, SAC, MEM-PPO, and MEM-SAC experiments in Q1. All Q2 methods use the SAC backbone to ensure a fair comparative analysis.

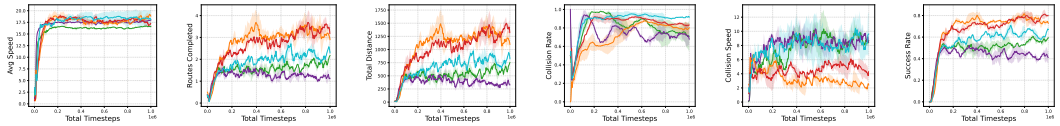
To comprehensively evaluate the baselines and our method, we utilize the quantitative metrics from VLM-RL benchmark. For driving efficiency assessment, we measure the average speed **(AS)**



(a) Performance comparison between MEM-SAC and SAC, categorized by Average Speed, Route Completion, Total Distance, Collision Rate, Collision Speed, and Success Rate.



(b) Performance comparison between MEM-PPO and PPO, categorized by Average Speed, Route Completion, Total Distance, Collision Rate, Collision Speed, and Success Rate.



(c) The performance comparison between MEM-SAC, VLM-RL, ChatScene, REvolve and LORD, categorized by Average Speed, Route Completion, Total Distance, Collision Rate, Collision Speed, and Success Rate.

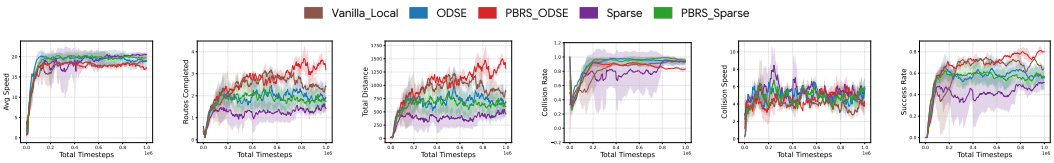
Figure 2: Performance comparison between our methods and the baselines. To reduce the influence of fluctuations at any single training step, we average each metric over the final 10% of training for each seed and report the mean and standard deviation across three seeds. The same setting is used for the remaining figures. Details are shown in Table 3.

maintained by the vehicle throughout episodes, the route completion **(RC)** which represents the number of successfully completed routes during one episode, and the total traveled distance **(TD)** which captures the cumulative distance covered by the vehicle during each episode. For safety performance evaluation, the fundamental collision rate **(CR)** measures the percentage of episodes containing collision events and collision speed **(CS)** measures the speed of the agent at the moment of the collisions. We further measure the success rate **(SR)**, defined as the percentage of **individual routes** successfully completed to the destination.

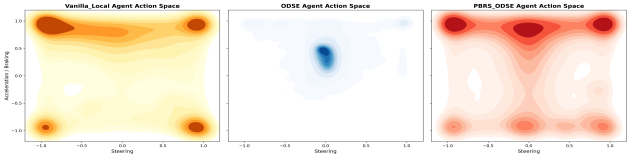
4.2 Main Results

In Figure 2a and Figure 2b, we compare the performance of our proposed methods (MEM-SAC and MEM-PPO) against their respective base learners (SAC and PPO) to answer Q1. For the PPO-based agents, vanilla PPO exhibits a low episode collision rate of 8%. However, this does not reflect superior safety but instead results from the agent's failure to drive. The agent frequently drives off the track, resulting in a near-zero route completion of 0.20 and a total distance of only 19.72m. In contrast, MEM-PPO successfully enables the agent to complete the task, achieving a route success rate of 51.8% and a route completion of 1.53. While active driving naturally leads to a higher collision rate (99%), the average speed increases from 16.27 km/h to 18.60 km/h, and the total distance reaches 539.59m. In the SAC-based comparison, MEM-SAC demonstrates consistent improvements across nearly all primary driving metrics, increasing the success rate from 68.5% to 78.4% and the route completion from 2.36 to 3.41. Although the average speed decreases slightly, the total distance improves from 877.40m to 1369.75m and the collision rate is notably reduced from 94% to 84%. These results indicate that our outcome-derived rewards significantly enhance the effectiveness and sample efficiency for both on-policy and off-policy end-to-end training.

For Q2, the results of the comparison between our method (MEM-SAC) and the baselines are shown in Figure 2c. Compared to state-of-the-art baselines, MEM-SAC achieves superior performance in task completion and driving efficiency. Most notably, MEM-SAC reaches the highest success rate of 78.4% and route completion of 3.41, surpassing the strongest baseline, VLM-RL, which achieves



(a) Performance comparison for the ablation of *Vanilla_Local*, *ODSE*, *PBRS_ODSE*, *Sparse* and *PBRS_Sparse*. Details are shown in Table 6.



(b) KDE action distribution of *Vanilla_Local*, *ODSE* and *PBRS_ODSE*. The x-axis denotes steering, where 0 corresponds to driving straight. The y-axis denotes throttle, where positive values represent acceleration and negative values represent braking.

Figure 3: Performance comparison for the ablation of ODSE and PBRS.

74.4% and 3.15, respectively. Other methods, such as LORD and ChatScene, exhibit lower success rates (61.0% and 56.7%) and cover significantly less distance per episode. Specifically, MEM-SAC maintains a total distance of 1369.75m, whereas VLM-RL and LORD cover 1183.94m and 766.30m. While the increased driving activity naturally leads to a higher collision rate than more conservative baselines like REvolve (73%) and ChatScene (74%), the collision speed of MEM-SAC (4.59 km/h) is substantially lower than that of LORD (8.71 km/h), ChatScene (8.81 km/h), and REvolve (9.01 km/h). These results demonstrate that our outcome-derived shaping signals provide strong guidance for training an effective and efficient end-to-end driving agent.

4.3 Ablation Study

4.3.1 Understanding the Roles of ODSE and PBRS

To understand why both ODSE and PBRS are needed, we compare five strategies: no reward shaping (*Vanilla_Local*), direct shaping with the SAC-ref ODSE (*ODSE*), PBRS with ODSE (*PBRS_ODSE*), direct shaping with a ± 1 sparse outcome signal (*Sparse*), and PBRS with the same sparse signal (*PBRS_Sparse*). For clarity, we use the SAC configuration detailed in Q1 as the basis for all the ablation experimental setups, including those in the remaining ablation studies.

As shown in Figure 3a, both *Sparse* and *PBRS_Sparse* remain inferior to *Vanilla_Local*, while *PBRS_ODSE* clearly improves training. Since ODSE is learned from the same terminal outcome information, this comparison shows that the benefit does not come from introducing the terminal outcomes themselves. Instead, ODSE makes this information available at intermediate states by estimating their relationship with eventual episode outcomes, providing substantially richer guidance than the original sparse signal.

However, these intermediate evaluations are not effective when directly added to the reward. Compared with *Vanilla_Local*, *ODSE* may reduce the success rate and total driving distance and also produces larger standard deviations. Directly adding ODSE causes its absolute state evaluations to accumulate throughout the trajectory, which can strongly modify the learning objective. PBRS instead incorporates the difference between consecutive ODSE values. As analyzed in Appendix A.2, the intermediate ODSE terms provide step-wise guidance while telescoping in the trajectory return, leaving only an explicitly characterized outcome-related terminal term in the learning objective. Therefore, ODSE provides informative evaluations of intermediate states, while PBRS provides a controlled way to use these evaluations during training.

To further examine how these different shaping strategies affect the learned behavior, we analyze the action distributions of *Vanilla_Local*, *ODSE*, and *PBRS_ODSE*. Although ODSE evaluates states rather than actions, changing the evaluation of successor states changes the actions preferred by the learned policy. We therefore record actions across 100 episodes on fixed routes in Town 2

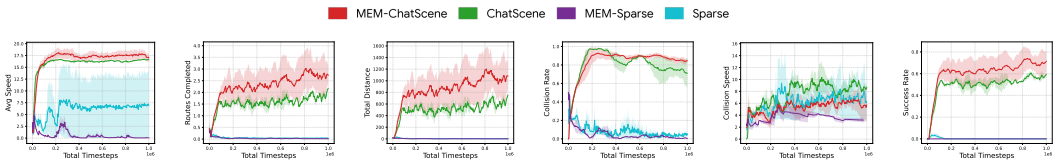
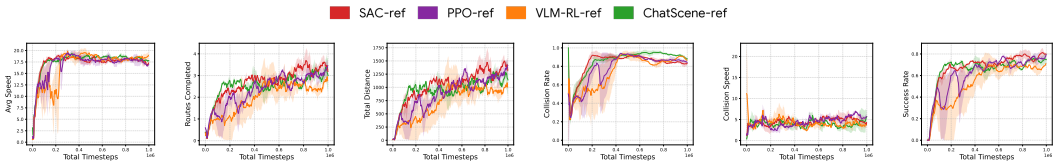
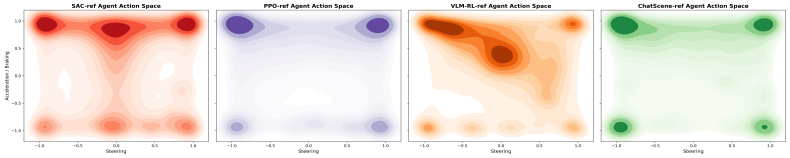


Figure 4: Performance comparison for the ablation of different local rewards. Details are shown in Table 4.



(a) Performance comparison for the ablation of reference pre-trained agents used for memory training. Details are shown in Table 5.



(b) KDE action distribution of agents guided by ODSE trained with four different references, SAC, PPO, VLM-RL and ChatScene. The x-axis denotes steering, where 0 corresponds to driving straight. The y-axis denotes throttle, where positive values represent acceleration and negative values represent braking.

Figure 5: Performance comparison for the ablation of reference agents.

and generate Kernel Density Estimate (KDE) distributions over steering and throttle. As shown in Figure 3b, *Vanilla_Local* concentrates many actions near the boundaries of the action space, while directly injecting ODSE produces an overly centralized distribution and reduces braking behavior. *PBRS_ODSE* retains the tendency toward more centralized driving while avoiding the stronger behavioral deviation caused by direct ODSE injection. We also share the KDE action distribution comparison of using different reference agents in Appendix A.5. In conclusion, these results suggest that ODSE has the ability to implicitly extract driving patterns through state evaluation, and incorporating PBRS helps avoid the policy deviations that direct ODSE injection might otherwise introduce.

4.3.2 Improving Different Local Rewards

In our main experiments, we utilize the local reward from VLM-RL and LORD. To verify our framework across diverse reward structures, we compare the original ChatScene local reward against its improved version (MEM-ChatScene), as well as a sparse local reward configured with a ± 1 signal identical to our ODSE generation rules against its improved version (MEM-Sparse).

As shown in Figure 4, when applied to the ChatScene reward, our framework improves the success rate from 56.7% to 69.1% and substantially increases the total distance. This confirms that outcome-derived signals can effectively augment diverse expert-designed reward structures. However, in the sparse reward configuration, both the vanilla baseline and MEM-Sparse converge to a "stopping" behavior, resulting in near-zero total distance. Notably, MEM-Sparse reaches this state significantly faster than the vanilla baseline, which is reflected in their respective average speeds. Since the sparse local reward is already fundamentally defined by task outcomes, the addition of outcome-derived shaping signals may not provide sufficient additional information to improve the agent's strategy. These results suggest that while our framework is effective at optimizing more informative reward structures, it does not overcome the inherent challenges of learning from strictly sparse local signals in complex driving tasks.

4.3.3 MEMORY TRAINING WITH DIFFERENT REFERENCE AGENTS

Because ODSE memory training can learn from various reference agents, a natural question is whether its effectiveness varies depending on the agent chosen. To investigate this, we conduct an ablation study by training the ODSE memory with four distinct reference agents, including the vanilla SAC (*SAC-ref*) and PPO (*PPO-ref*) agents from Q1, and agents trained with the ChatScene local reward (*ChatScene-ref*) and VLM-RL shaped rewards (*VLM-RL-ref*) from Q2.

As shown in Figure 5a, our method maintains high performance and stability across reference agents with different learning algorithms and reward structures. It remains effective when referencing agents trained with ChatScene and VLM-RL, reaching success rates of 74.1% and 72.3%. Notably, deriving ODSE from a **highly suboptimal** PPO reference agent (with a success rate of only 0.4%) still achieves a success rate of 76.1% and a total driving distance of 1259.32 m. This shows that reference-policy performance itself need not be high, since both successful and failed trajectories provide outcome supervision. Following the KDE analysis in Section 4.3.1, we apply the same analysis to agents guided by ODSEs learned from different reference policies. Figure 5b shows that the resulting policies exhibit distinct action distributions. *SAC-ref* displays a relatively centralized driving pattern, *VLM-RL-ref* also exhibits centralized driving but uses less throttle, and *ChatScene-ref* brakes more frequently than the other references. These results show that ODSEs learned from different reference agents can induce different driving behaviors.

Direct reuse of reference trajectories. Since SAC is off-policy, we also test mixing the same 5×10^5 reference transitions into its replay batches. Reference replay reduces collisions and improves SAC, providing independent support for ODSE's premise that reference trajectories contain transferable information. However, ODSE achieves higher SR, RC, and TD and also applies to on-policy learners such as PPO. The details are shown in Appendix A.6.

5 DISCUSSION AND CONCLUSION

Regarding the scope of our empirical validation, all experiments are conducted within the CARLA-based VLM-RL benchmark. This reliance raises potential questions regarding how broadly our proposed framework might transfer to other simulators or real-world driving datasets. However, to the best of our knowledge, VLM-RL remains the only end-to-end autonomous driving benchmark specifically tailored for evaluating and comparing diverse reward designs. Other benchmarks in the autonomous driving domain are typically not equipped for this specific type of reward-centric evaluation, and may function as non-interactive simulators (e.g., NAVSIM Dauner et al. (2024)), which cannot support the closed-loop, online policy exploration required by RL approaches. Consequently, while our evaluation scope is constrained to this single benchmark, it currently represents the most appropriate and rigorous environment available for validating our reward shaping strategy.

The primary limitation of our method lies in understanding the training dynamics induced by ODSE-based reward shaping. Appendix A.2 characterizes the effect of PBRS difference structure at the trajectory level, showing that the intermediate ODSE terms telescope and leave an explicitly characterized outcome-related terminal term in the learning objective. However, this analysis does not reveal how different ODSE evaluations affect the optimization process. We therefore investigate this question empirically. Our ablation studies show that ODSE can identify and evaluate distinct driving patterns exhibited by different reference agents. We think a formal theoretical analysis of the relationship between the scale and variance of our shaping signals and optimization stability remains a promising direction for future research.

In this paper, we propose a reward shaping method utilizing Outcome-Derived State Evaluation (ODSE) to bridge the gap between semantic approximations and physical outcomes. Our method introduces a learned memory module to leverage experiences from reference agents, generating state evaluations from trajectory-level terminal outcome supervision. By integrating these evaluations through the potential-difference formulation of Potential-Based Reward Shaping (PBRS), we derive a dense shaping signal that provides step-wise guidance while introducing an explicitly characterized outcome term into the learning objective. Our experiments on the VLM-RL benchmark demonstrate that our approach outperforms state-of-the-art expert-designed and model-generated reward functions. We believe our exploration of the relationship between states and terminal outcomes advances the development of effective autonomous driving policies.

AI Disclosure Section

In this work, we used generative AI tools to edit author-written text for grammar, clarity and readability. We also used generative AI tools to assist with the presentation and interpretation of an additional baseline experiment. We did not use generative AI tools to generate the research idea or hypotheses, develop the proposed method, write the code, perform experiments or quantitative analyses, formulate mathematical claims or proofs, or draft the initial manuscript. All technical ideas, methodology, code, experimental design and execution, results, and initial paper content were produced by the authors and all AI-assisted edits were manually reviewed against the underlying experiments and manuscript. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Reproducibility Statement

We provide the implementation and experimental details required to reproduce our results. The main experiments are conducted on the VLM-RL benchmark using CARLA 0.9.13 and Stable-Baselines3, with the training settings, evaluation metrics, and reference-data collection procedure described in Section 4.1. The ODSE generation and reward-shaping procedures are specified in Section 3 and Appendix A.1, including pseudo-code for ODSE memory training and policy training with ODSE. Additional implementation details, hyperparameter analyses, cross-town experiments, and supplementary ablations are provided in the Appendix. We further provide an anonymous code repository at `https://anonymous.4open.science/r/ODSE-3281/README.md`.

References

Zhong Cao, Shaobing Xu, Xinyu Jiao, Huei Peng, and Diange Yang. Trustworthy safety improvement for autonomous driving using reinforcement learning. *Transportation research part C: emerging technologies*, 138:103656, 2022.

Shuo Feng, Haowei Sun, Xintao Yan, Haojie Zhu, Zhengxia Zou, Shengyin Shen, and Henry X Liu. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*, 615(7953): 620–627, 2023.

Tong Nie, Jian Sun, and Wei Ma. Exploring the roles of large language models in reshaping transportation systems: A survey, framework, and roadmap. *Artificial Intelligence for Transportation*, 1:100003, 2025.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

Jiawei Zhang, Chejian Xu, and Bo Li. Chatscene: Knowledge-enabled safety-critical scenario generation for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15459–15469, 2024.

Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5068–5078, 2021.

Xu Han, Qiannan Yang, Xianda Chen, Zhenghan Cai, Xiaowen Chu, and Meixin Zhu. Autoreward: Closed-loop reward design with large language models for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2024.

Rishi Hazra, Alkis Sygkounas, Andreas Persson, Amy Loutfi, and Pedro Zuidberg Dos Martires. Revolve: Reward evolution with large language models using human feedback. *arXiv preprint arXiv:2406.01309*, 2024.

Kate Baumli, Satinder Baveja, Feryal Behbahani, Harris Chan, Gheorghe Comanici, Sebastian Flennerhag, Maxime Gazeau, Kristian Holsheimer, Dan Horgan, Michael Laskin, et al. Vision-language models as a source of rewards. *arXiv preprint arXiv:2312.09187*, 2023.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*, 2023.

Xin Ye, Feng Tao, Abhirup Mallik, Burhaneddin Yaman, and Liu Ren. Lord: Large models based opposite reward design for autonomous driving. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5072–5081. IEEE, 2025.

Zilin Huang, Zihao Sheng, Yansong Qu, Junwei You, and Sikai Chen. Vlm-rl: A unified vision language models and reinforcement learning framework for safe autonomous driving. *Transportation Research Part C: Emerging Technologies*, 180:105321, 2025.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer, 1999.

Sumedh Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies. *Advances in Neural Information Processing Systems*, 36:55681–55693, 2023.

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Hongwei Xie, Bing Wang, Guang Chen, Dingkang Liang, and Xiang Bai. Minddrive: A vision-language-action model for autonomous driving via online reinforcement learning. *arXiv preprint arXiv:2512.13636*, 2025.

Dawood Wasif, Terrence J Moore, Chandan K Reddy, and Jin-Hee Cho. Drivemind: A dual-vlm based reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:2506.00819*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of machine learning research*, 22(268):1–8, 2021.

Letian Wang, Jie Liu, Hao Shao, Wenshuo Wang, Ruobing Chen, Yu Liu, and Steven L. Waslander. Efficient reinforcement learning for autonomous driving with parameterized skills and priors, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

A Appendix

A.1 Implementation Details

A.1.1 ODSE Memory Structure

In Table 1, we give the architecture of the ODSE memory. The input of the memory is a state vector with N dimensions. Initially, a Token Projection layer transforms each of the N state dimensions into a 256-dimensional vector, which is then combined with a Feature ID embedding to maintain the

Table 1: The architecture of the ODSE memory

MODULE	SHAPE
TOKEN PROJECTION	$(N, 1) \rightarrow (N, 256)$
FEATURE ID EMBEDDING	$(N, 256)$
TRANSFORMER ENCODER	$(N, 256)$
GLOBAL MEAN POOLING	(256)
LINEAR 1	$(256, 512)$
RELU	
LINEAR 2	$(512, 1)$

identity of each feature. These tokens are processed by a Transformer Encoder followed by mean pooling to generate latent representations. Finally, we use a two-layer MLP network (Linear1 + ReLU + Linear2) to output the predicted scalar ODSE value for the input state. During the update process, we use Adam as the optimizer and MSELoss as the loss function.

A.1.2 ODSE MEMORY TRAINING

Algorithm 1 Memory Training for ODSE Generation

Input: E: Environment, $\mathcal{A}_{pre}$: Pre-trained Agent, M: Neural Memory
Parameter: N_{steps}: Total collection steps
Output: M: Trained ODSE Memory
for $step = 1$ to N_{steps} **do**
  $\mathcal{A}_{pre}$ interacts with E to generate state s_t and reward r_t
  Add s_t to current trajectory τ
  if episode ends at step T **then**
    Observe outcome-related terminal score $sc(\tau)$
    Assign $sc(\tau)$ to each $s_t \in \tau$
    Train M with the s_t-$sc(\tau)$ pairs from τ
    Clear τ
  end if
end for
return M

Algorithm 1 details the data collection and training procedure for the ODSE neural memory (M). During this process, a pre-trained reference agent ($\mathcal{A}_{pre}$) interacts with the environment (E) for a predefined total of N_{steps} steps, sequentially appending each encountered state (s_t) to an ongoing trajectory buffer (τ). When an episode terminates at step T, an outcome-related terminal score ($sc(\tau)$) is observed to characterize the episode's performance. Crucially, this single terminal score is retroactively assigned to every individual state $s_t \in \tau$ to form a dataset of state-score pairs. The neural memory M is then trained on these pairs, utilizing its network architecture to map the individual states to their ultimate episode outcomes. After the memory is updated, the trajectory buffer is cleared to begin the next episode. This iterative cycle of interaction, retroactive scoring, and memory training continues until the total step limit is reached, yielding the final trained ODSE memory.

A.1.3 REWARD SHAPING WITH ODSE

Algorithm 2 details the reward shaping procedure during the training of the learning agent. The pre-trained memory M serves as a fixed evaluator to augment the local rewards from the environment. For each transition (s_t, a_t, r_t, s_{t+1}), we calculate the state value differential. By querying the ODSE values for both the current and successor states, we compute $\delta(s_t)$, which is then incorporated into the original reward signal to improve the training process.

A.2 THEORETICAL ANALYSIS OF ODSE-BASED PBRS

Using the definition in Equation 3, the shaped discounted return at step t is expressed as:

Algorithm 2 Reward Shaping with ODSE

Input: E: Environment, M: Pre-trained ODSE Memory
Parameter: ϵ: Revision Coefficient
Output: Revised transition $(s_t, a_t, r'_t, s_{t+1})$
while Target Agent interacts with E **do**
  Observe current transition (s_t, a_t, r_t, s_{t+1})
  Query evaluations $\tilde{V}(s_t) \leftarrow M(s_t)$ and $\tilde{V}(s_{t+1}) \leftarrow M(s_{t+1})$
  if s_t is non-terminal **then**
    $\delta(s_t) \leftarrow \gamma\tilde{V}(s_{t+1}) - \tilde{V}(s_t)$
  else
    $\delta(s_t) \leftarrow 0$
  end if
  $r'_t \leftarrow r_t + \epsilon \cdot \delta(s_t)$
  return $(s_t, a_t, r'_t, s_{t+1})$
end while

$$\begin{aligned} G'_t = \sum_{l=0}^{T-t} \gamma^l r'_{t+l} &= \sum_{l=0}^{T-t} \gamma^l (r_{t+l} + \epsilon \cdot \delta(s_{t+l})) \\ &= \sum_{l=0}^{T-t} \gamma^l r_{t+l} - \epsilon \cdot \sum_{l=0}^{T-t} -\gamma^l \delta(s_{t+l}) \\ &= G_t - \epsilon \cdot (\tilde{V}(s_t) - \gamma^{T-t}\tilde{V}(s_T)), \end{aligned} \tag{4}$$

where $\delta(s_t)$ is the potential-based shaping term for state s_t, $\tilde{V}(s_t)$ is the ODSE of s_t, and G_t is the unshaped discounted return. Therefore, the revised policy gradient $\nabla_\theta J'(\theta)$ becomes:

$$\begin{aligned} \nabla_\theta J'(\theta) &= \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G'_t\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t\right] \\ &\quad - \epsilon \cdot \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \tilde{V}(s_t)\right] \\ &\quad + \epsilon \cdot \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \gamma^{T-t} \tilde{V}(s_T)\right]. \end{aligned} \tag{5}$$

The second term in Equation 5 involves $\tilde{V}(s_t)$, which is fully determined by the current state s_t, so it is a standard value baseline (Williams, 1992) and contributes zero to the expectation:

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\, \tilde{V}(s_t)\right] = \mathbb{E}\left[\tilde{V}(s_t)\, \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) \mid s_t\right]\right] = 0. \tag{6}$$

The third term involves the terminal evaluation. Since the memory is trained to predict the terminal score, and terminal states carry their outcomes most directly, we have $\tilde{V}(s_T) \approx sc(\tau)$, so this term is ϵ times the policy gradient of a discount-weighted outcome objective. Putting the two together, the revised policy gradient is:

$$\nabla_\theta J'(\theta) = \nabla_\theta J(\theta) + \epsilon \cdot \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, \gamma^{T-t} sc(\tau)\right]. \tag{7}$$

To further understand what the learner optimizes in total, we set $t = 0$ in Equation 4 and take the expectation, which gives:

$$J'(\theta) = J(\theta) + \epsilon \cdot \mathbb{E}_{\tau \sim \pi_\theta} \left[\gamma^T \tilde{V}(s_T) \right] - \epsilon \cdot \mathbb{E} \left[\tilde{V}(s_0) \right]. \tag{8}$$

The last term depends only on the initial states and not on the policy, so it does not affect learning. The only change to what the learner optimizes is therefore an ϵ-weighted task outcome: ϵ controls its strength, and $\epsilon \to 0$ recovers the original objective. However, this added term is not the reason for the improvement by itself, since adding a similar outcome term directly as a sparse reward does not improve training (see Section 4.3.1). Our method spreads the same objective into dense per-step signals through the intermediate evaluations $\tilde{V}(s_t)$, and Equation 8 guarantees that this dense guidance adds nothing to the expected gradient beyond the outcome term. Overall, this section certifies that the step-wise guidance is safe and fully characterized, while the empirical benefit requires the step-wise form itself (see Section 4.3.3 and Section 4.3.1). How the intermediate evaluations of a particular reference shape the training dynamics remains an open question, which we discuss in Section 5.

A.3 Parameter Analysis

A.3.1 Shaping Coefficient

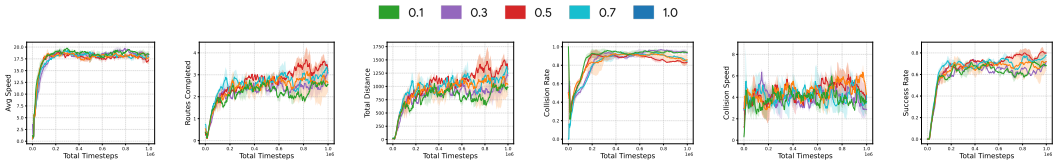


Figure 6: Parameter analysis of 5 different shaping coefficients, including 0.1, 0.3, 0.5, 0.7, 1.0. Details are shown in Table 7.

In Section 3.4, we introduce the shaping coefficient ϵ to control the intensity of the reward shaping. If the coefficient is too large, the shaping term may overwhelm the primary environmental rewards, potentially leading to sub-optimal convergence. Conversely, if ϵ is too small, the guidance remains insufficient to effectively bridge the gap between sparse terminal outcomes and states. Therefore, we conduct an analysis of 5 different coefficients: *0.1*, *0.3*, *0.5*, *0.7*, and *1.0*. As shown in Figure 6, the agent achieves the most robust performance with $\epsilon = 0.5$, reaching a peak success rate of 80% and the highest route completion. This demonstrates that $\epsilon = 0.5$ provides the optimal balance between empirical guidance and environment-driven exploration.

A.3.2 Memory Training Steps

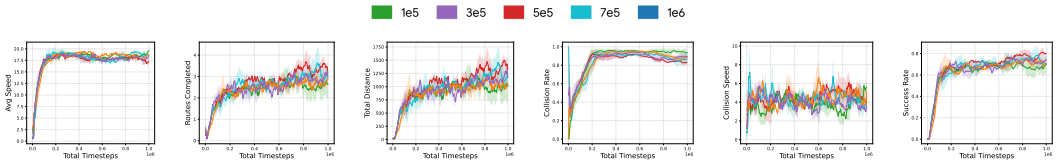


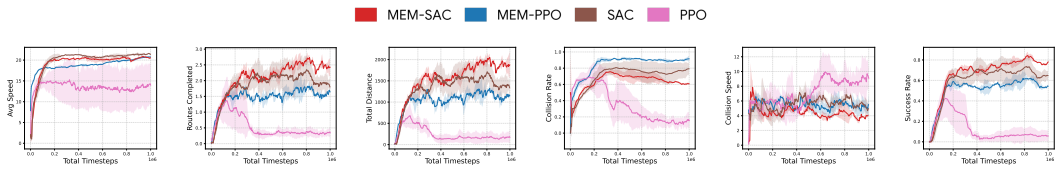
Figure 7: Parameter analysis of the total training steps of ODSE memory. Details are shown in Table 8.

We also analyze the total training steps for the ODSE memory module to determine the optimization duration required for stable state evaluation. We evaluate five different durations: *1e5*, *3e5*, *5e5*, *7e5*, and *1e6* steps. As shown in Figure 7, our method demonstrates high robustness to this parameter, as all tested durations of at least *1e5* steps yield acceptable results with success rates exceeding 0.73. This indicates that the memory module effectively captures the essential relationship between states

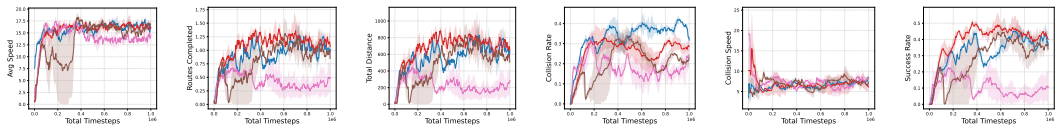
and terminal outcomes across a wide range of training intervals. In practice, training for *1e5* steps requires approximately 2 hours, while *5e5* steps takes 10 hours. Training for *5e5* steps is acceptable for both time consumption and effectiveness, so we select it as the total steps for ODSE memory training in our main experiments.

A.4 Cross-Town Experiments

A.4.1 Cross-Town Training



(a) Performance comparison between MEM-SAC and SAC, MEM-PPO and PPO trained in Town 1, categorized by Average Speed, Route Completion, Total Distance, Collision Rate, Collision Speed, and Success Rate. Details are shown in Table 9.



(b) Performance comparison between MEM-SAC and SAC, MEM-PPO and PPO trained in Town 3, categorized by Average Speed, Route Completion, Total Distance, Collision Rate, Collision Speed, and Success Rate. Details are shown in Table 10.

Figure 8: The performance comparison of agents trained in Town 1 and Town 3.

To verify that our training framework is robust across diverse environments, we conduct additional training experiments in Town 1 and Town 3, expanding upon the main experiments conducted in Town 2. Figure 8 illustrates the performance of our methods (MEM-SAC and MEM-PPO) against their vanilla counterparts (SAC and PPO). Consistent with our observations in Town 2, vanilla PPO struggles to learn effective driving policies in both new environments. In contrast, MEM-PPO successfully stabilizes the training process and achieves competitive performance across both maps. Similarly, MEM-SAC consistently matches or outperforms vanilla SAC across crucial metrics such as Route Completion (**RC**), Total Distance (**TD**), and Success Rate (**SR**). These results empirically demonstrate that the improvements provided by our approach in boosting performance are robust and map-agnostic.

A.4.2 Cross-Town Evaluation

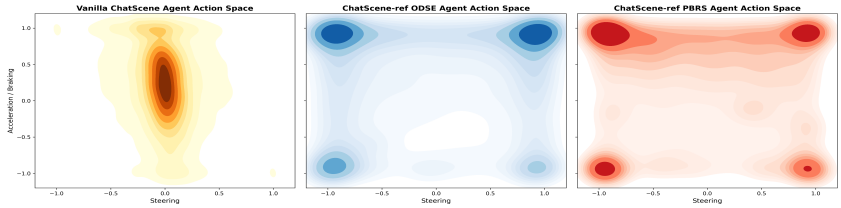
Table 2: Evaluation of MEM-SAC across different town maps. For each town map, we test the MEM-SAC agent for 100 routes and repeat this procedure three times. Note the route completion (RC) here shows the average route completion rate across 100 routes.

Model	AS (km/h)	RC	CR	CS (km/h)	SR
Town 1	20.56± 3.75	0.99± 0.00	0.03± 0.01	1.14± 0.66	0.97± 0.01
Town 2	18.33± 1.40	0.95± 0.04	0.21± 0.03	2.02± 0.54	0.79± 0.03
Town 3	18.02± 0.46	0.92± 0.02	0.11± 0.02	1.02± 0.08	0.89± 0.02
Town 4	21.00± 2.33	0.81± 0.11	0.32± 0.03	0.21± 0.02	0.68± 0.03
Town 5	22.37± 4.35	0.92± 0.03	0.19± 0.02	1.83± 0.96	0.81± 0.02

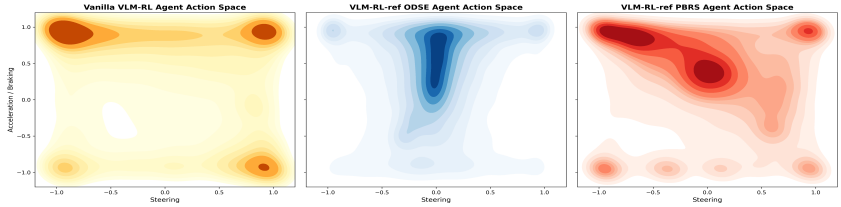
Beyond the robustness of the training process, we also assess the zero-shot transferability of the final learned policy. To this end, we evaluate the Town 2-trained MEM-SAC agent from Q1 across all town maps. As shown in Table 2, the agent achieves excellent performance in environments like Town 1, reaching a success rate of 97%, and maintains consistent route completion rates above 92%

in Towns 3 and 5. While the metrics for Town 4 are comparatively lower, the agent still demonstrates capable driving, yielding a 68% success rate and a route completion of 0.81, alongside a notably low collision speed of 0.21 km/h. Overall, these results confirm that the driving policy learned by our agent remains highly robust when transferred from its training environment to entirely unseen map layouts.

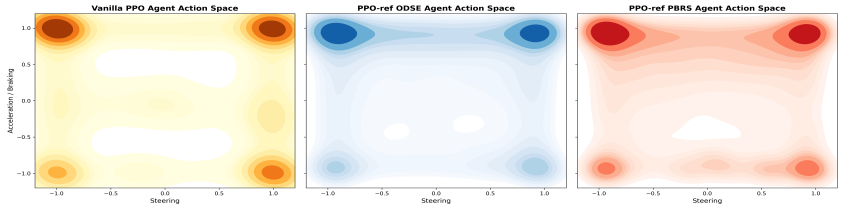
A.5 Supplementary for ODSE and PBRS



(a) KDE action distribution of *Vanilla ChatScene*, *ChatScene-ref ODSE*, *ChatScene-ref PBRS_ODSE*. The x-axis denotes steering (0 corresponds to driving straight), and the y-axis denotes throttle (positive values represent acceleration, negative values represent braking).



(b) KDE action distribution of *Vanilla VLM-RL*, *VLM-RL-ref ODSE*, *VLM-RL-ref PBRS_ODSE*. The x-axis denotes steering (0 corresponds to driving straight), and the y-axis denotes throttle (positive values represent acceleration, negative values represent braking).



(c) KDE action distribution of *Vanilla PPO*, *PPO-ref ODSE*, *PPO-ref PBRS_ODSE*. The x-axis denotes steering (0 corresponds to driving straight), and the y-axis denotes throttle (positive values represent acceleration, negative values represent braking).

Figure 9: Supplementary results for the ablation of ODSE and PBRS.

In this section, we share the KDE action distribution comparison for *ODSE* and *PBRS_ODSE* of *PPO-ref*, *ChatScene-ref*, and *VLM-RL-ref* from Section 4.3.3. From Figure 9a and Figure 9b, we can easily see that the ODSEs learned from ChatScene and VLM-RL are also extracting different driving patterns to guide the training with PBRS, just like the one learned from vanilla SAC. However, the three distributions in Figure 9c are quite similar to each other. After checking the recordings of the three agents, we find that while the vanilla PPO agent frequently drives off the track, both the *PPO-ref ODSE* agent and the *PPO-ref PBRS* agent can perform successful driving. This indicates that although varying action distributions correlate with distinct driving behaviors, similar distributions do not inherently guarantee comparable driving outcomes. This discrepancy underscores the need for a formal theoretical analysis to fully elucidate the underlying mechanics of the ODSE + PBRS framework.

A.6 Direct Reuse of Reference Trajectories

A natural alternative to learning an ODSE from the reference trajectories is to directly reuse these trajectories during policy optimization. This is possible for an off-policy learner such as SAC, whose

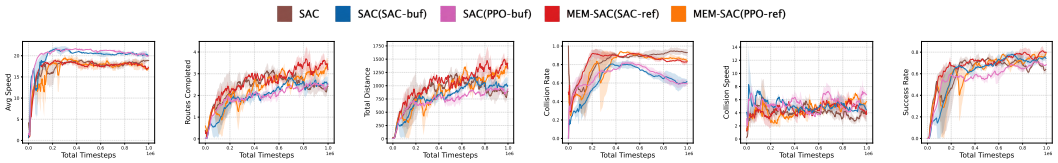


Figure 10: Training performance of SAC with direct reference-transition reuse compared with vanilla SAC and ODSE-based reward shaping. SAC(SAC-buf) and SAC(PPO-buf) mix reference transitions into each SAC minibatch with a fixed ratio of $p = 0.25$, using reference datasets collected by the SAC and PPO agents, respectively. All methods follow the Q1 experimental setting. Details are shown in Table 11.

updates can be performed using transitions generated by a different behavior policy. We therefore construct a direct reference-replay baseline to examine whether the improvements obtained from ODSE can instead be achieved by simply exposing the learner to the same reference experience.

Setup. We maintain a reference buffer separately from the learner's online replay buffer. The reference buffer contains 5×10^5 environment steps collected by the same pre-trained reference agents used to train the ODSE memories in Q1, so the replay baseline and our method use the same reference source and reference-data budget. Each stored sample contains the reference agent's state, action, and transition information. Rather than retaining the reward received by the reference agent, we recompute the reward using the learning agent's local reward function, ensuring consistency with the learner's reward definition. At each SAC gradient update, a fraction $p = 0.25$ of the minibatch is sampled from the reference buffer and the remaining $1 - p$ is sampled from the learner's online replay buffer. We evaluate reference buffers generated by both the SAC and PPO reference agents from Q1, denoted as **SAC(SAC-buf)** and **SAC(PPO-buf)**, respectively. Since this mechanism relies on off-policy replay, it is evaluated only with SAC.

As shown in Figure 10, directly reusing the reference transitions provides a meaningful learning signal. Both reference-buffer variants reduce the collision rate from 0.94 for vanilla SAC to 0.60, and SAC(SAC-buf) also improves the success rate from 0.685 to 0.738. This confirms that the reference trajectories are informative even though they are off-policy with respect to the learning agent.

However, collision rate alone does not fully characterize successful driving in this benchmark. Episodes can also terminate because the vehicle leaves the road or remains stationary, and therefore a lower collision rate does not necessarily imply more complete driving. This is also observed in our PPO experiments, where a policy that largely fails to drive can exhibit a deceptively low collision rate. Accordingly, CR should be considered together with the task-completion metrics. On the completion metrics, the two mechanisms show a different empirical profile. Both replay variants improve upon vanilla SAC to some extent, but MEM-SAC achieves larger gains in route completion, total traveled distance, and success rate. With the SAC reference, for example, direct replay reaches an SR of 0.738, RC of 2.53, and TD of 997 m, whereas MEM-SAC reaches 0.784, 3.41, and 1370 m, respectively. A similar difference is observed with the PPO reference data.

The two approaches therefore exploit the same reference experience in different forms. Reference replay transfers the transitions themselves and lets the learner's critic evaluate these transitions under the learner's reward function. In contrast, ODSE first extracts an outcome-related state evaluation from the reference trajectories and transfers this learned evaluation to states visited by the learning agent through reward shaping. Empirically, direct transition reuse yields a larger reduction in collision rate in this experiment, whereas ODSE shaping yields larger improvements in the completion-related metrics. These observations suggest that the two mechanisms may capture complementary aspects of the reference experience rather than being interchangeable.

Finally, reference replay is specific to off-policy learners that can train on transitions generated by another behavior policy. ODSE-based reward shaping does not have this restriction and can also be applied to on-policy learners. In Q1, applying ODSE shaping to PPO increases the success rate from 0.004 to 0.518. Thus, even when direct transition reuse is available for SAC, reward shaping provides a more general mechanism for transferring outcome-related information across learning algorithms.

A.7 Experiment Details

The details of the figures are provided on the subsequent pages.

Table 3: Performance comparison between our method and baselines. For each seed, each metric is averaged over the final 10% of training. We report the mean and standard deviation across seeds. The best results are marked in **bold**. PPO is excluded from this marking since it rarely drives.

Model	AS (km/h) ↑	RC ↑	TD (m) ↑	CR ↓	CS (km/h) ↓	SR ↑
SAC	**18.62± 0.50**	2.36± 0.22	877.40± 84.64	0.94± 0.04	3.61± 0.62	0.685± 0.015
PPO	16.27± 0.04	0.20± 0.04	19.72± 2.45	**0.08± 0.04**	10.63± 1.40	0.004± 0.001
MEM-SAC	17.53± 0.18	**3.41± 0.17**	**1369.75± 37.60**	0.84± 0.02	4.59± 0.92	**0.784± 0.021**
MEM-PPO	18.60± 0.13	1.53± 0.03	539.59± 22.05	0.99± 0.00	3.88± 0.25	0.518± 0.012
ChatScene	16.60± 0.13	1.86± 0.09	621.48± 18.92	0.74± 0.10	8.81± 0.13	0.567± 0.011
REvolve	17.77± 0.01	1.17± 0.11	343.04± 32.52	0.73± 0.06	9.01± 0.41	0.418± 0.021
VLM-RL	18.59± 0.63	3.15± 0.36	1183.94± 133.83	0.79± 0.06	**2.34± 0.40**	0.744± 0.022
LORD	18.30± 1.74	2.14± 0.09	766.30± 41.28	0.92± 0.00	8.71± 2.18	0.610± 0.006

Table 4: Performance comparison for the ablation of local rewards. For each seed, each metric is averaged over the final 10% of training. We report the mean and standard deviation across seeds.

Local Reward	AS (km/h) ↑	RC ↑	TD (m) ↑	CR ↓	CS (km/h) ↓	SR ↑
MEM-ChatScene	17.53± 0.82	2.70± 0.73	1025.95± 327.74	0.85± 0.03	5.87± 2.17	0.691± 0.103
ChatScene	16.60± 0.13	1.86± 0.09	621.48± 18.92	0.74± 0.10	8.81± 0.13	0.567± 0.011
MEM-Sparse	0.05± 0.05	0.02± 0.01	0.56± 0.50	0.01± 0.00	3.23± 0.32	0.000± 0.000
Sparse	6.92± 6.83	0.04± 0.03	3.76± 2.97	0.04± 0.02	6.90± 3.79	0.000± 0.000

Table 5: Performance comparison for the ablation of reference agents used for memory training. For each seed, each metric is averaged over the final 10% of training. We report the mean and standard deviation across seeds.

Reference Agent	AS (km/h) ↑	RC ↑	TD (m) ↑	CR ↓	CS (km/h) ↓	SR ↑
SAC	17.53± 0.18	3.41± 0.17	1369.75± 37.60	0.84± 0.02	4.59± 0.92	0.784± 0.021
PPO	17.42± 0.56	3.10± 0.31	1259.32± 130.42	0.86± 0.02	5.88± 0.45	0.761± 0.036
ChatScene	18.06± 0.28	3.07± 0.24	1164.69± 73.70	0.91± 0.01	4.45± 2.02	0.741± 0.019
VLM-RL	18.55± 0.48	2.51± 0.11	954.11± 74.81	0.90± 0.01	3.90± 0.22	0.723± 0.030

Table 6: Performance comparison for the ablation of ODSE and PBRS. For each seed, each metric is averaged over the final 10% of training. We report the mean and standard deviation across seeds.

SHAPING TERM	AS (KM/H) ↑	RC ↑	TD (M) ↑	CR ↓	CS (KM/H) ↓	SR ↑
VANILLA_LOCAL	18.62± 0.50	2.36± 0.22	877.40± 84.64	0.94± 0.04	3.61± 0.62	0.685± 0.015
ODSE	19.09± 1.79	1.91± 0.54	678.59± 245.27	0.94± 0.05	4.84± 1.01	0.570± 0.093
PBRS_ODSE	17.53± 0.18	3.41± 0.17	1369.75± 37.60	0.84± 0.02	4.59± 0.92	0.784± 0.021
SPARSE	20.46± 0.61	1.44± 0.21	481.77± 105.42	0.94± 0.05	5.35± 0.87	0.494± 0.058
PBRS_SPARSE	19.71± 0.52	1.83± 0.34	652.18± 144.06	0.97± 0.01	5.24± 0.98	0.578± 0.062

Table 7: Parameter analysis of 5 different shaping coefficients. For each seed, each metric is averaged over the final 10% of training. We report the mean and standard deviation across seeds.

ϵ	AS (KM/H) ↑	RC ↑	TD (M) ↑	CR ↓	CS (KM/H) ↓	SR ↑
0.1	18.44± 0.47	2.56± 0.03	980.73± 74.85	0.94± 0.01	3.59± 1.35	0.683± 0.011
0.3	18.45± 0.62	3.09± 0.41	1201.93± 159.21	0.93± 0.02	2.84± 0.44	0.711± 0.005
0.5	17.53± 0.18	3.41± 0.17	1369.75± 37.60	0.84± 0.02	4.59± 0.92	0.784± 0.021
0.7	17.89± 0.08	3.06± 0.14	1244.27± 37.15	0.85± 0.02	4.17± 0.35	0.774± 0.033
1.0	17.84± 1.36	2.63± 0.42	1137.87± 207.27	0.86± 0.05	4.27± 0.09	0.721± 0.056

Table 8: Parameter analysis of the total training steps of ODSE memory. For each seed, each metric is averaged over the final 10% of training. We report the mean and standard deviation across seeds.

STEPS	AS (KM/H) ↑	RC ↑	TD (M) ↑	CR ↓	CS (KM/H) ↓	SR ↑
1E5	19.62± 0.29	2.63± 0.71	1015.46± 336.02	0.91± 0.04	5.59± 0.85	0.738± 0.07
3E5	18.28± 0.21	2.92± 0.39	1227.79± 209.19	0.88± 0.04	3.34± 0.88	0.742± 0.052
5E5	17.53± 0.18	3.41± 0.17	1369.75± 37.60	0.84± 0.02	4.59± 0.92	0.784± 0.021
7E5	18.59± 0.51	3.07± 0.29	1204.02± 109.99	0.86± 0.04	3.57± 0.75	0.766± 0.023
1E6	18.75± 0.29	2.84± 0.08	1143.12± 8.12	0.88± 0.02	5.12± 1.13	0.754± 0.01

Table 9: Performance comparison between MEM-SAC, SAC, MEM-PPO and PPO trained in Town 1. For each seed, each metric is averaged over the final 10% of training. We report the mean and standard deviation across seeds. The best results are marked in **bold**.

MODEL	AS (KM/H) ↑	RC ↑	TD (M) ↑	CR ↓	CS (KM/H) ↓	SR ↑
MEM-SAC	20.54± 0.12	**2.42± 0.29**	**1872.33± 306.94**	0.61± 0.01	**4.05± 0.05**	**0.784± 0.013**
SAC	**21.38± 0.28**	1.80± 0.19	1336.93± 147.87	0.80± 0.07	4.95± 1.21	0.642± 0.042
MEM-PPO	20.74± 0.15	1.65± 0.08	1147.44± 45.51	0.91± 0.03	5.51± 1.37	0.558± 0.01
PPO	13.97± 5.03	0.35± 0.16	171.11± 140.96	**0.15± 0.07**	9.07± 1.36	0.051± 0.043

Table 10: Performance comparison between MEM-SAC, SAC, MEM-PPO and PPO trained in Town 3. For each seed, each metric is averaged over the final 10% of training. We report the mean and standard deviation across seeds. The best results are marked in **bold**.

MODEL	AS (KM/H) ↑	RC ↑	TD (M) ↑	CR ↓	CS (KM/H) ↓	SR ↑
MEM-SAC	15.98± 1.24	**1.06± 0.21**	**648.82± 112.84**	0.29± 0.02	**6.13± 1.18**	**0.417± 0.051**
SAC	**16.85± 1.34**	0.82± 0.01	504.44± 19.28	**0.23± 0.02**	7.30± 0.85	0.33± 0.015
MEM-PPO	16.071± 0.54	0.98± 0.02	615.51± 0.54	0.31± 0.01	8.46± 1.83	0.376± 0.023
PPO	14.38± 0.12	0.49± 0.28	286.13± 170.80	0.24± 0.05	8.17± 0.21	0.113± 0.111

Table 11: Comparison between direct reuse of reference transitions and ODSE-based reward shaping in the Q1 setting. Reference-replay methods mix a fixed fraction $p = 0.25$ of reference transitions into each SAC minibatch. Both approaches use the same 5×10^5-step reference datasets.

MODEL	AS (KM/H) ↑	RC ↑	TD (M) ↑	CR ↓	CS (KM/H) ↓	SR ↑
SAC	18.62±0.50	2.36±0.22	877±85	0.94±0.04	3.61±0.62	0.685±0.015
SAC(SAC-BUF)	20.24±0.60	2.53±0.10	997±65	**0.60±0.06**	4.86±0.67	0.738±0.006
SAC(PPO-BUF)	20.24±0.43	2.46±0.09	917±58	**0.60±0.04**	6.26±1.08	0.695±0.017
MEM-SAC(SAC-REF)	17.53±0.18	**3.41±0.17**	**1370±38**	0.84±0.02	4.59±0.92	**0.784±0.021**
MEM-SAC(PPO-REF)	17.42±0.56	3.10±0.31	1259±130	0.86±0.02	5.88±0.45	0.761±0.036