Scaling Curriculum Learning For Autonomous Driving

Cevahir Koprulu*[1], David Paz[2], Feng Tao[2], Yuliang Guo[2], Xinyu Huang[2]
Ufuk Topcu[1], Liu Ren[2]
[1]The University of Texas at Austin, [2]Bosch Center for AI, North America

Abstract

Batched simulators for autonomous driving have recently enabled training reinforcement learning (RL) agents at scale, encompassing thousands of traffic scenarios and billions of interactions within a matter of days. Although such high-throughput feeds RL algorithms faster than ever, their sample-efficiency has not kept pace: As the standard training scheme, domain randomization uniformly samples scenarios, thereby consuming a vast number of interactions on cases that contribute little to learning. Curriculum learning offers a remedy by adaptively prioritizing scenarios that matter most to policy improvement. We present CL4AD, the first integration of curriculum learning into batched autonomous driving simulators by framing scenario selection as an unsupervised environment design problem. We introduce utility functions that shape curricula based on success rates and the realism of the agent's behavior, in addition to existing regret-estimation functions. Large-scale experiments in GPUDRIVE demonstrate that curriculum learning achieves a 99% success rate a billion steps earlier than domain randomization, reducing wall-clock time by 77%, and outperforms heuristic curricula with static and dynamic attributes, with only one exception at the largest scale. An ablation under limited compute shows that curriculum learning improves sample efficiency by 67%. We also investigate how utility functions behave at scale, and how prioritized scenarios evolve during training. We release an implementation of CL4AD in GPUDRIVE.

1 Introduction

Batched simulators for autonomous driving (AD) have recently empowered sample-inefficient but effective reinforcement learning (RL) algorithms by enabling training for billions of interactions within days [10, 22]. These simulators achieve such scale by training RL agents on hundreds to thousands of scenarios in parallel through self-play [38], where a single policy controls all vehicles, taking millions of actions per second. Agents trained on GPUDRIVE [22] using the Waymo Open Motion Dataset (WOMD) [14] generalize to unseen test cases in less than a day. GIGAFLOW [10], further scales self-play to 1.6 billion kilometers of simulated driving within 10 days, producing generalist driving policies that outperform benchmark-specific agents on CARLA [12], nuPlan [7], and Waymax [16] without any training on these benchmarks.

Despite advances in high simulation throughput, training RL agents in batched driving simulators remains sample-inefficient due to a standard training strategy: uniform scenario sampling, i.e., domain randomization (DR). This approach wastes interactions on scenarios that are either too easy to provide a sufficient learning signal or too difficult for the current policy to make progress on. Curriculum learning (CL) offers a remedy by adaptively prioritizing scenarios that contribute the most to policy improvement [27]. In particular, curriculum learning has successfully fulfilled that promise in multiple large-scale RL domains. For example, [5] demonstrate that scaling meta-RL

*Corresponding author: cevahir.koprulu@utexas.edu

Preprint.

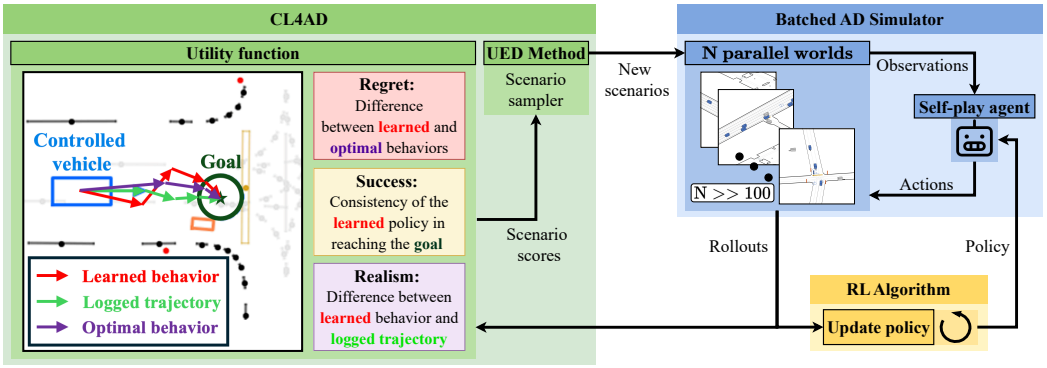


Figure 1: CL4AD integrates a UED method into a batched AD simulator to adaptively prioritize traffic scenarios based on three types of utility functions: regret, success, and realism. **(blue)** A self-play agent interacts with sampled scenarios across concurrent worlds. **(yellow)** An RL algorithm of choice updates the policy using collected rollouts in said scenarios. **(green)** CL4AD scores scenarios based on a chosen utility function and updates its sampler via the integrated UED method.

with automated curricula yields agents capable of human-timescale adaptation across thousands of procedurally generated environments. [42] introduce curricula for open-ended environments with infinitely many tasks, showing that curriculum learning enables faster and broader skill acquisition.

Inspired by the success of CL in large-scale RL, we introduce *Curriculum Learning for Autonomous Driving* (CL4AD) (see Figure 1), the first integration of automated curricula into a batched AD simulator. We frame scenario selection as an unsupervised environment design problem (UED), and equip a UED method with utility functions that adaptively shape training. Therefore, the curricula prioritize scenarios at the edge of the agent's capabilities. Across experiments, we show that CL accelerates RL training by hundreds of millions of steps compared to DR and heuristic curricula.

Our key contributions are three-fold:

- We present **CL4AD**, the first integration of curriculum learning methods from unsupervised environment design to the scale of GPU-accelerated, self-play simulators for AD, and provide an implementation in an open-source batched AD simulator, GPUDRIVE.
- We propose **novel utility functions** based on realism, and a safety-critical success criterion that balances robustness and safety, exposing the gap between optimal and realistic driving.
- We conduct a **large-scale empirical study** 1) showing that CL4AD accelerates RL training by up to a billion interactions compared to DR and outperforms heuristic curricula; 2) investigating how prioritized scenarios evolve during training; 3) analyzing the correlation among utility functions and performance metrics; 4) describing how multi-agent self-play creates a persistent learning frontier; 5) demonstrating that reward optimality and behavioral realism are distinct axes; and 6) providing practitioner guidelines for curriculum learning in batched AD simulators.

2 Related Work

Autonomous driving simulators have enabled RL to train self-driving agents in multiple ways: Waymax [16] and Nocturne [41] use traffic scenarios from open-source driving datasets such as WOMD [14], whereas CARLA [13] is not data-driven, and Metadrive enables procedural scenario generation as well as integration of real driving data. Attempts to scale RL for AD have led to batched simulators such as Waymax, GPUDRIVE [22], and GIGAFLOW [10], which significantly increase data throughput for RL algorithms but still rely on random scenario generation/sampling.

Curriculum learning for RL accelerates learning optimal policies by sequencing different configurations of the environment [27]. Automated curriculum generation studies goal-conditioned domains [4, 15, 40], contextual settings [23, 24, 34], and more popularly UED [11]. Across many settings, a curriculum requires a signal that measures the contribution of a task to policy improvement, e.g., learning progress, i.e., the change in an agent's competence, which [30] estimates over a continuous parameter space and [21] measures as the change in success probability. UED methods can generate levels through trained teachers or by sampling free parameters. PAIRED [11] trains a teacher that

emits level parameters maximizing regret against an antagonist, and RE-PAIRED [18] stabilizes this game by replaying high-regret levels. ACCEL [29] instead mutates the parameters of high-regret levels in a replay buffer and curates the mutants that score highly. PLR [19], as one of the earlier UED approaches, neither trains a teacher nor mutates parameters; instead, it scores and replays levels it has already encountered. PLR has also shown evidence of scalability in meta RL [5, 17], and open-ended environments [42]. Beyond individual methods, libraries such as DCD [18], minimax [20], and Syllabus [39] implement curricula for batched or asynchronous training, though they target procedurally generated environments outside AD.

Curriculum learning for AD aims to speed up training self-driving policies via RL, e.g., ScenarioNet [25], which unifies heterogenous data for simulation, showcases benefits of heuristic-based curricula. Similarly, [2, 3] propose a multi-stage curriculum learning method for CARLA, making the number of agents, their initial positions, or weather conditions incrementally more difficult. In contrast to manual curricula, [31] develops an automated method for urban intersections. Recently, [6] and [1] have demonstrated that UED methods, RE-PAIRED and ACCEL, respectively, accelerate training AD agents in CARLA. However, there has been no investigation into whether CL can scale with the high throughput and scenario diversity enabled by batched simulators such as GPUDRIVE, which trains RL agents on real-world scenarios from the Waymo Open Motion Dataset (WOMD) [14]. Other curriculum strategies for AD include adversarial scenario generation [43], continual policy adaptation with individualized curricula [28], and VLM-guided safety-critical curriculum design [37]. These works train single-ego agents on a smaller scale, in contrast to CL4AD, which enables training with real-world datasets across thousands of concurrent self-play agents in hundreds of parallel worlds.

3 Background

We model a traffic scenario as a *partially observable stochastic game* (POSG) [6], to accommodate the multi-agent nature of driving. Then, we frame curriculum learning for autonomous driving as *unsupervised environment design*, and describe how to measure the utility of traffic scenarios.

3.1 Traffic scenarios as partially observable stochastic games

Definition 3.1. A *POSG* is a tuple $\mathcal{G} = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, I, \gamma \rangle$, where $\mathcal{N} = [\mathrm{N}]$ is the set of agents with $\mathrm{N} \in \mathbb{Z}^+$, $\mathcal{S}$ is the state space, $\mathcal{A} = \prod_{i=1}^{\mathrm{N}} \mathcal{A}_i$ and $\mathcal{O} = \prod_{i=1}^{\mathrm{N}} \mathcal{O}_i$ are the joint action and observation spaces. $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the stochastic dynamics, i.e, the probability of transitioning from state $\mathbf{s} \in \mathcal{S}$ to state $\mathbf{s}' \in \mathcal{S}$ given joint action $\mathbf{a} \in \mathcal{A}$. $Z : \mathcal{S} \times \mathcal{A} \to \Delta(Z)$ determines the probability of observing $\mathbf{o} = (\mathbf{o}_1, \mathbf{o}_2, \cdots, \mathbf{o}_N) \in \mathcal{O}$ in state $\mathbf{s}$ taking joint action $\mathbf{a}$. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^N$ is the reward function, i.e., $R(\mathbf{s}, \mathbf{a}) = (R_i(\mathbf{s}, \mathbf{a}))_{i \in [\mathrm{N}]}$ where $R_i(\mathbf{s}, \mathbf{a}) \in \mathbb{R}$ is the reward for agent $i \in \mathcal{N}$. $I \in \Delta(\mathcal{S})$ is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor.

A policy $\pi_i : \mathcal{O}_i \to \Delta(\mathcal{A}_i)$ describes the behavior of agent i in POSG $\mathcal{G}$. The value function for π_i is the expected cumulative discounted rewards over a horizon of H steps, i.e., $V(\pi_i) = \mathbb{E}_{T,Z}\left[\sum_{t=0}^{\mathrm{H}-1} \gamma^t R_i(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{s}_o \sim I, \mathbf{a}_t = (\mathbf{a}_{j,t})_{j \in \mathcal{N}}\right]$ where $\mathbf{a}_{j,t} \sim \pi_j(\mathbf{o}_{j,t})$. Agent i aims to find an optimal policy π_i^*, which maximizes its value $V(\pi_i)$ in POSG $\mathcal{G}$.

In a traffic scenario modeled as $\mathcal{G}$, consider π_i as a policy that controls vehicle i. The road layout, traffic rules, and collision dynamics in a scenario specify the dynamics T. Initial state $\mathbf{s}_0 \sim I$ consists of the initial positions of all vehicles, pedestrians, cyclists, etc. Observation $\mathbf{o}_{i,t}$ of vehicle i at time $t \in [\mathrm{H}]$ is what the controller perceives about the surroundings based on its sensors as well as specific attributes, e.g., the type of vehicle, its velocity, acceleration, etc. The reward $r_i = R_i(\mathbf{s}_t, \mathbf{a}_t)$ can incentivize the policy to reach a goal location, stay within lanes, and avoid collisions. To model multiple traffic scenarios, we formalize them as an *underspecified* POSG (UPOSG).

Definition 3.2. An *underspecified POSG* $\mathcal{G}^\Theta = \langle \Theta, \mathcal{N}^\Theta, \mathcal{S}, \mathcal{A}^\Theta, \mathcal{O}^\Theta, T^\Theta, Z^\Theta, R^\Theta, I^\Theta, \gamma \rangle$ models a set of POSGs through parameters $\theta \in \Theta$ that determine all attributes of a POSG $\theta \in \Theta$ depending on its agents, such as the dynamics $T^\Theta : \mathcal{S} \times \mathcal{A}^\Theta \times \Theta \to \Delta(\mathcal{S})$.

Consider scenarios $\Theta = \{\theta_m\}_{m \in [\mathrm{M}]}$ in WOMD, where $\mathrm{M} \approx 100,000$. A scenario θ_m may correspond to an urban intersection or a highway, with varying speed limits, number of vehicles, etc. In practice, θ_m is merely an identification number, i.e., $\theta_m \in [\mathrm{M}]$, hence it is underspecified.

3.2 Unsupervised Environment Design

UED [11] aims to generate a sequence of *levels*[2], i.e., scenarios $\theta \in \Theta$ in the case of AD, to accelerate learning a policy that generalizes across all levels. One solution to UED is a level generator $\Lambda : \Pi \to \Delta(\Theta)$ that produces a distribution over the set of levels Θ given a policy $\pi \in \Pi$. A level generator Λ maximizes a utility function $U(\pi, \theta)$ measuring the contribution of θ for improving π.

Domain randomization, i.e., uniformly sampling levels throughout the training, is the default way of training an RL agent where the utility is constant for each level, namely, $U(\pi, \theta) = \mathrm{C} \in \mathbb{R}, \forall \theta \in \Theta$. UED methods primarily differ in their utility functions of choice. There are two common categories of utility functions: regret and success-based.

Regret, i.e., the difference between the expected discounted return of the current policy and the optimal one, enables prioritizing the easiest levels that the agent cannot currently solve [11]. More formally, a regret-based utility is $U^{\text{Regret}}(\pi, \theta) = V^{\theta}(\pi^*_{\theta}) - V^{\theta}(\pi)$, where π^*_{θ} is an optimal policy in level θ, i.e., a policy collecting the maximum expected discounted return $V^{\theta}(\pi^*_{\theta})$. However, as the optimal expected discounted return or the optimal policy for each level is rarely available, UED methods estimate regret in various ways. [19] propose learning potential, i.e., *average magnitude of the generalized advantage estimate* (AMGAE) [35], (a) in Figure 2, as a utility function that estimates regret over a single episode. Here, $\delta_{k.n} = r_{k,n} + \gamma V^{\theta,\pi}(\mathbf{o}_{k+1,n}) - V^{\theta,\pi}(\mathbf{o}_{k,n})$ is the temporal difference error at timestep k of agent $n \in [\mathrm{N}_{\theta}]$. $V^{\theta,\pi}(\mathbf{o}_{k,n}) = \mathbb{E}_{T^{\Theta}, Z^{\Theta}}\left[\sum_{t=k}^{\mathrm{H}-k-1} \gamma^{t-k} R^{\Theta}(\mathbf{s}_{t,n}, \mathbf{a}_{t,n}, \theta) | \mathbf{a}_{t,n} \sim \pi(\mathbf{o}_{t,n})\right]$ is the expected discounted return of agent n in $\mathbf{s}_k$ on level θ, and λ is the discount factor for GAE. Alternatively, [18] and [29] employ *positive value loss* (PVL), (b) in Figure 2. As PVL uses the bootstrapped value target to compute the temporal difference error, [18] also propose *maximum Monte Carlo* (MaxMC), (c) in Figure 2, which instead utilizes the highest return obtained by agent n on level θ to mitigate potential bias issues, where $\mathrm{R}^{\theta,n}_{\text{max}}$ is the maximum discounted return achieved in level θ so far during training.

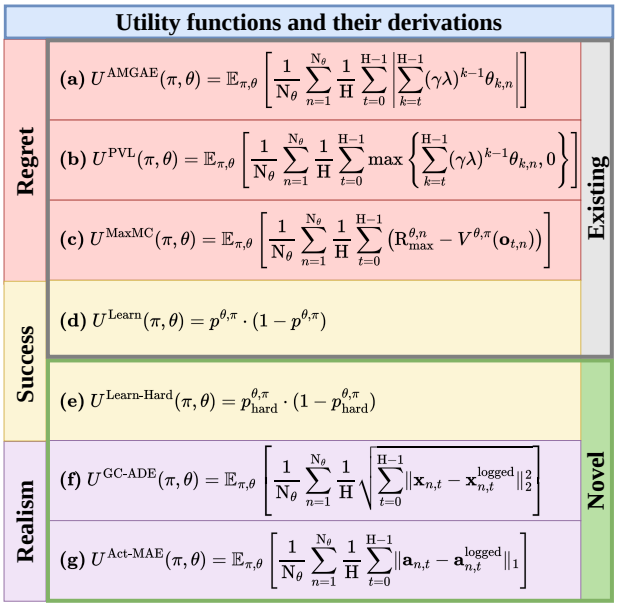


Figure 2: Utility functions: CL4AD adapts existing **(a-d)** utility functions as well as proposes **(e-g)** novel ones.

Success-based utility functions address settings where a level θ is considered solved for agent n when a policy π reaches a goal state $\mathbf{s} \in \mathcal{S}^{\theta,n}_{\text{Goal}} \subset \mathcal{S}$. Such utility functions use the success rate $p^{\theta,\pi}$, i.e., the fraction of controlled agents that reach a goal state in an episode of level θ under policy π, $p^{\theta,\pi} = \frac{1}{\mathrm{N}}\sum_{n\in\mathrm{N}} \mathbb{1}[\exists t \in [\mathrm{H}] : \mathbf{s}_t \in \mathcal{S}^{\theta,n}_{\text{Goal}}]$. Inspired by [40], [33] propose *Sampling for Learnability* (SFL), along with *learnability*, (d) in Figure 2, a utility function corresponding to the variance of a Bernoulli distribution with parameter $p^{\theta,\pi}$, namely, how inconsistently policy π solves θ across all controlled agents. [33] argues that, in sparse-reward settings, regret-based utility functions exhibit low correlation with success rates, as regret-based utility functions become noisy in such settings, causing inaccurate identification of the learning frontier. In AD, rewards commonly occur after sparse events, such as goal completion or collisions; thus, learnability becomes a viable alternative.

4 Curriculum Learning for Autonomous Driving at Scale

CL4AD scales UED to batched AD simulators. CL4AD integrates variants of a UED method, *prioritized level replay* (PLR) [19], which lays the foundation for approaches such as Robust PLR

[2]As *level* is the common term in the UED literature to describe θ, we use it interchangeably with *scenario*.

Algorithm 1 **C**urriculum **L**earning for **A**utonomous **D**riving (CL4AD)

Input: Set of training scenarios Θ^{train}

Parameters: Replay rate d, Staleness coefficient ρ, temperature β, utility function U, max buffer size B^{max}, total number of iterations T^{train}, scenario sampling interval T^{sce}, policy update interval T^{pol}, number of worlds W

Output: Final policy π_ϕ

$\mathcal{B} \leftarrow (), \mathcal{D} \leftarrow () \; t \leftarrow 0, l \leftarrow 0, \pi_\phi \leftarrow \pi_{\phi_0}$ ▷ *Reset scenario/experience buffers, iterators, and policy*

while $t < \text{T}^{\text{train}}$ **do**

  if $0 \equiv t \mod \text{T}^{\text{sce}}$ **then**

    $l \leftarrow l + 1$ ▷ *Increment sampling iteration*

    $(\theta_w)_{w=1}^{\text{W}}, \mathcal{B} \leftarrow \text{SAMPLEFROMCURRICULUM}(\mathcal{B}, \Theta^{\text{train}}, l)$ ▷ *Sample scenarios for worlds*

  end if

  $\mathcal{D}_t = \{\{\mathbf{o}_{n,w}, \mathbf{a}_{n,w}, \mathbf{o}'_{n,w}, r_{n,w}, e_{n,w}\}_{n \in [\text{N}_{\theta_w}]}\}_{w \in [\text{W}]}$ ▷ *Record experiences over a single step*

  $\mathcal{B} \leftarrow \text{UPDATECURRICULUM}(\mathcal{D}_t, U, \mathcal{B})$ ▷ *Update curriculum with the scores of terminated scenarios*

  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_t$ ▷ *Update experience buffer with new interactions*

  if $0 \equiv t \mod \text{T}^{\text{pol}}$ **then**

    $\pi, \mathcal{D} \leftarrow \Phi(\mathcal{D})$ ▷ *Update self-play policy via RL algorithm Φ, and reset the experience buffer $\mathcal{D}$*

  end if

  $t \leftarrow t + |\mathcal{D}_t|$ ▷ *Update training iteration*

end while

[18], REPAIRED, ACCEL, and SFL. CL4AD scales them up for a batched AD simulator on four axes: **(1)** concurrent simulation of hundreds of scenarios, **(2)** tracking tens of agents per scenario, **(3)** training in tens of thousands of scenarios, and **(4)** for billions of steps.

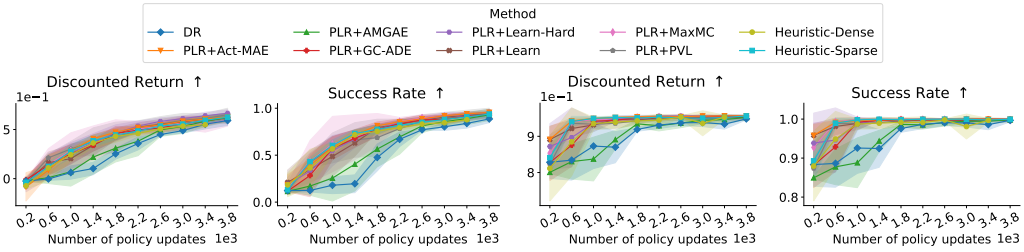
CL4AD efficiently scales via asynchronous episode management. CL4AD tracks the behavior of all controlled agents in all concurrent scenarios to compute their utility, which captures the expected collective behavior of the self-play policy. In GPUDRIVE, where we implement CL4AD, simulated scenarios come from real-world datasets, and each scenario has a specific horizon H due to the nature of the logged data. Since scenarios have different horizons and agents terminate at different times, CL4AD processes episode terminations asynchronously: it computes the utility of a scenario as soon as all its agents terminate, updates the score by averaging over the episodes of that scenario since the last sampling call, and frees the rollout buffer for the next episode. This keeps memory usage proportional to the number of active episodes rather than the total number of episodes between curriculum sampling steps. Wall-clock measurements show that curriculum updates account for approximately 1% of total training time (see Section B).

CL4AD operates over large training sets by adapting PLR's sampling mechanism, which has two parts: uniformly sampling scenarios from the training set Θ^{train}, and replaying levels from a rolling buffer $\mathcal{B}$. At the beginning of the training, PLR uniformly randomly samples scenarios, and scores them based on a utility function U. Then, it adds scenarios with the highest scores to its buffer. Subsequently, PLR makes a random decision with probability d to sample unseen levels in Θ^{train} or seen levels from the buffer via a distribution based on their scores and staleness, namely,

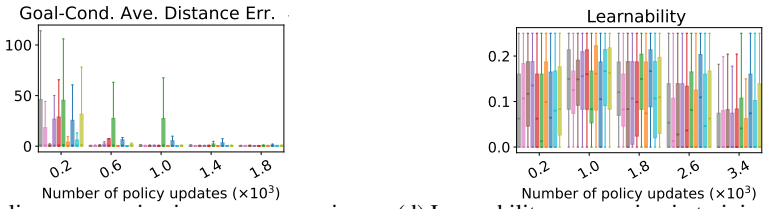
$$\mathbb{P}_{\text{replay}}(\theta_i|\mathcal{B}, U, l) = (1-\rho) \cdot \mathbb{P}_{\text{utility}}(\theta_i|\mathcal{B}, U) + \rho \cdot \mathbb{P}_{\text{staleness}}(\theta_i|\mathcal{B}, l), \tag{1}$$

where $\mathbb{P}_{\text{utility}}(\theta_i|\mathcal{B}, U)$ emphasizes levels with higher ranks with respect to their scores, and $\mathbb{P}_{\text{staleness}}(\theta_i|\mathcal{B}, l)$ assigns a higher likelihood for levels that have not been sampled for longer (see Section B). This distribution aims to prevent the scores of seen levels from becoming off-policy, as they may remain in the buffer for a while without being sampled.

CL4AD introduces three novel utility functions, (e-g) in Figure 2: *learnability-hard* $U^{\text{Learn-hard}}$, *goal-conditioned average distance error* (GC-ADE) $U^{\text{GC-ADE}}$, and *action mean absolute error* (Act-MAE) $U^{\text{Act-MAE}}$. $U^{\text{Learn-hard}}$ is a success-based utility function that, in contrast to U^{Learn}, utilizes the rate at which agents reach their goals without colliding or going off-road in scenario θ via self-play policy π. Since episodes do not terminate on collision or off-road events, an agent can complete its goal unsafely, which U^{Learn} counts as a success and $U^{\text{Learn-hard}}$ does not. $U^{\text{Learn-hard}}$ therefore applies learnability to a success criterion that AD works use, as it captures both robustness and safety [10]. $U^{\text{GC-ADE}}$ and $U^{\text{Act-MAE}}$ are realism-based utility functions that compute the distance between the positions and actions of RL agents and the logged trajectories, respectively, evaluating the plausibility of behavior [7, 16, 8].



(a) Evaluation in training scenarios (b) Evaluation in unseen scenarios



(c) Realism progression in unseen scenarios (d) Learnability progression in training scenarios

Figure 3: Case 1: Training with 1000 scenarios. **(top)** We demonstrate performance progression in **(a)** training, and **(b)** test partitions, where bold markers indicate the mean and the shaded area covers one standard deviation across three independent runs. **(bottom)** Box plots illustrate quartiles of two utility functions: **(c)** $U^{\text{GC-ADE}}$ and **(d)** U^{Learn} progression in test/training splits.

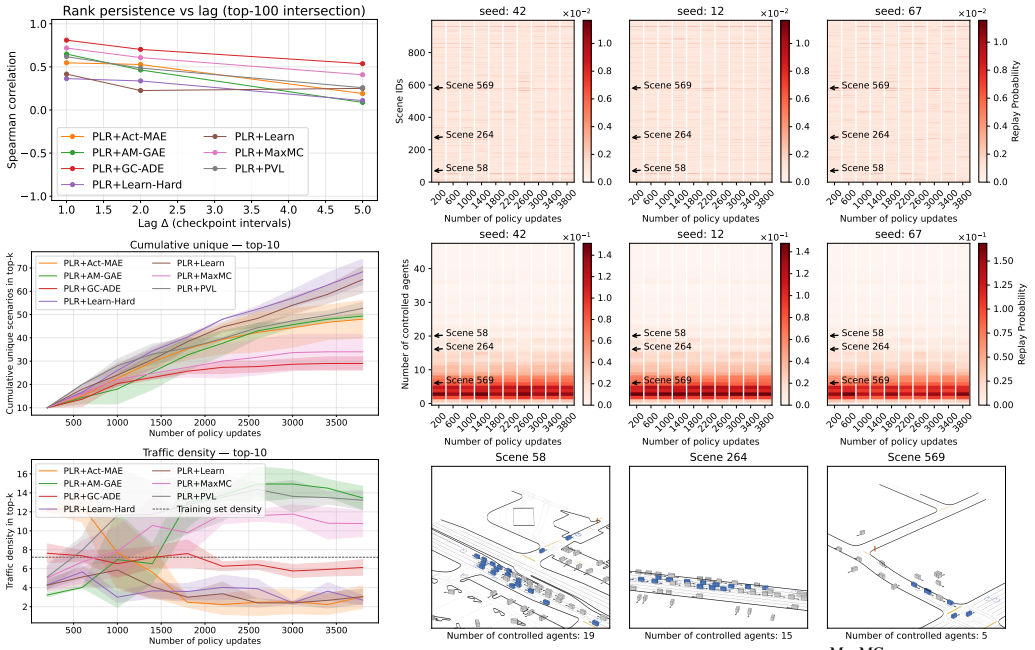
Algorithm 1 illustrates CL4AD. At the beginning of the training, we initialize self-play policy π_ϕ, and reset scenario and experience buffers $\mathcal{B}$ and $\mathcal{D}$, as well as the training and scenario sampling iterations, t and l, respectively (Line 1). Until training iteration reaches T^{train}, CL4AD first checks if it is time to sample new scenarios based on its replay buffer $\mathcal{B}$ (Line 3-5). If so, PLR samples new scenarios, and CL4AD sets them to concurrently simulated worlds. Note that PLR only keeps B^{max} highest ranking scenarios in the buffer for sampling. Then, the self-play policy π_ϕ takes a step in all scenarios, and $\mathcal{D}_t$ records them (Line 7). CL4AD updates the curriculum buffer using the utility of terminated scenarios (Line 8). Finally, an RL algorithm updates the policy using the experience buffer $\mathcal{D}$ (Line 9-12) every T^{pol} steps. We refer the reader to Section B for more details.

5 Experimental Results

We implement CL4AD in GPUDRIVE [22] and conduct experiments using traffic scenarios from WOMD [14]. We assess performance via return and success rates, safety via collision and off-road rates, and realism via GC-ADE and metrics from the Waymo Open Sim Agents Challenge (WOSAC)[26]. We train RL agents using self-play PPO following [22, 9], with sparse rewards for goal completion, collisions, and going off-road. Note that an episode does not terminate upon collision/off-road. Qualitatively, we analyze curriculum evolution, the correlation among utility functions and performance metrics, and how multi-agent self-play shapes the learning frontier. We compare DR against four heuristic curricula and seven PLR variants combined with the utility functions in Figure 2. Heuristic-Dense/Sparse prioritize scenarios with high/low vehicle counts; Heuristic-Fast/Far prioritize by descending mean velocity and pairwise inter-agent distance, respectively. Note that we adopt U^{PVL} and U^{MaxMC} from Robust PLR and U^{Learn} from SFL, rather than the algorithms themselves. See Section C for more details on experiments.

5.1 Can CL4AD accelerate learning?

PLR variants significantly accelerate learning, outperforming DR and heuristic curricula in both sample-efficiency and realism. Figures 3a and 3b shows that PLR, with all utility functions except U^{AMGAE}, achieves the highest returns and success rates in training scenarios. Figure 5b evidences that, in test scenarios, PLR achieves 99% success rate a billion steps earlier than DR, reducing wall-clock time by 77%. Compared with Heuristic-Sparse/Dense, PLR accelerates training to achieve the same success rate by 40% and 66%, respectively. Note that PLR with U^{AMGAE} outperforms DR with a small margin in terms of return. PLR also yields realistic policies faster than



(a) Score ranking evolution. (b) Progression of $\mathbb{P}_{\text{replay}}$ for U^{MaxMC}

Figure 4: Case 1: How CL4AD guides scenario selection. **(a)** We analyze score rankings: (top) Spearman rank correlation at increasing lags ($\Delta = 1, 2, 5$) for the top-100 scenarios, where lower correlation indicates faster reshuffling; (middle) cumulative number of unique scenarios appearing in the top-10 over training; (bottom) mean number of controlled agents in the top-10 scenarios, with the dashed line indicating the training set average. **(b)** $\mathbb{P}_{\text{replay}}$ progression of PLR with U^{MaxMC}: (top) the evolution of $\mathbb{P}_{\text{replay}}$, darker segments are scenarios with higher likelihood, (middle) replay distribution with respect to the number of controlled agents, (bottom) three frequently replayed scenarios.

DR (see Figure 3c). Lastly, Figure 3d illustrates that approaches that converge early obtain high learnability early on and achieve the lowest learnability fastest at the end.

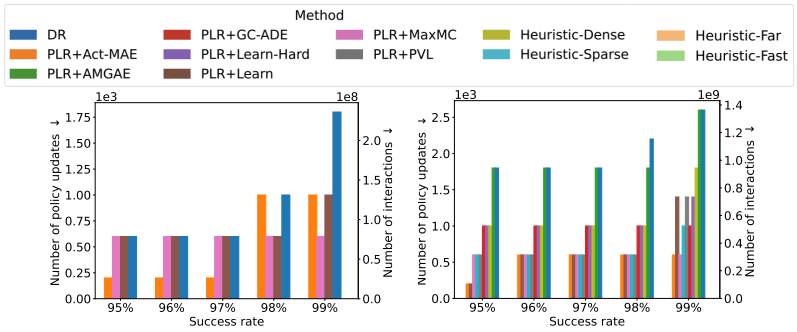
5.2 How does CL4AD guide scenario selection?

CL4AD actively reshuffles score rankings, though the rate varies across utility functions. The rank persistence plot (top row in Figure 4a) shows the Spearman rank correlation between score rankings. $U^{\text{GC-ADE}}$ and U^{MaxMC} are the most rank-persistent, maintaining correlations of $0.4 - 0.55$ at $\Delta = 5$. Other utility functions, particularly success-based and $U^{\text{Act-MAE}}$, decorrelate more rapidly, reaching $0.1 - 0.25$ at $\Delta = 5$. Cumulative number of unique scenarios (middle row) corroborates this: PLR with $U^{\text{GC-ADE}}$ and U^{MaxMC} see 29 and 34, respectively, unique scenarios enter their top-10 over the full training run, while success-based and regret-based functions cycle through $48 - 70$.

Different utility functions favor qualitatively distinct instances. The progression of traffic density across the top-10 ranked scenarios (bottom row of Figure 4a) reveals that utility functions guide the curriculum to qualitatively distinct scenarios. Regret-based functions increasingly prioritize scenarios with traffic density above the dataset average. For example, Figure 4b shows the replay distribution progression of PLR with U^{MaxMC}: dense scenarios such as Scenes 58 and 264 maintain high probability across runs, consistent with this mechanism. Success-based functions shift toward scenarios with low density as training progresses: scenarios with few controlled agents have the highest Bernoulli variance, since a single failure has a larger impact on the success rate.

5.3 Is CL4AD effective under limited compute?

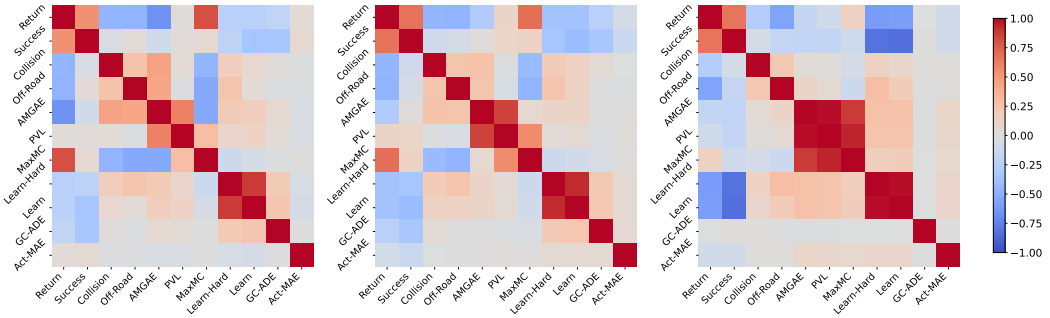
CL4AD remains effective under strict computational constraints, outperforming DR in time to success. Figure 5a illustrates results from an ablation study using a GPU with significantly smaller memory, reducing the number of worlds W and the experience buffer $\mathcal{D}$ (see Section D). Although a



(a) Ablation: 1,000 scenarios (b) Case 1: 1,000 scenarios

(c) Case 2: 10,000 scenarios (d) Case 3: 80,000 scenarios

Figure 5: Success progression in unseen test scenarios: **(a)** ablation study under compute constraints, **(b-d)** training in 1,000, 10,000, and 80,000 scenarios, respectively.



(a) Case 1: 1,000 scenarios (b) Case 2: 10,000 scenarios (c) Case 3: 80,000 scenarios

Figure 6: Pearson correlation between utility functions and performance metrics, i.e, success, collision, and off-road rates: Results from training in **(a)** 1,000 **(b)** 10,000, and **(c)** 80,000 scenarios.

smaller buffer results in more frequent policy updates, this setup takes about four times as long in wall-clock time and limits the diversity of scenarios used for updates. DR needs fewer interactions than the regular set-up, yet PLR reaches a 99% success rate 67% faster than DR.

5.4 How does CL4AD scale up the dataset?

CL4AD continues to deliver sample-efficiency gains as the number of training scenarios increases to tens of thousands. We train self-play agents in (case 2) 10,000 and (case 3) 80,000 scenarios from WOMD. Figure 5c demonstrates that PLR reduces the number of interactions needed to reach a 99% success rate by over 55%, when combined with U^{MaxMC} and $U^{\text{Act-MAE}}$ in case 2. Similarly, Figure 5d shows that PLR improves sample-efficiency by 72% when combined with U^{Learn} in case 3. Here, only Heuristic-Dense matches PLR, whereas the other heuristics offer no advantage.

5.5 Do utility functions correlate with each other and performance metrics?

Figure 6 presents correlations between utility functions and performance metrics across all training cases. We include policies from multiple checkpoints to capture agents at varying stages of learning.

Regret-based functions differ in how they handle sparse rewards, and their approximations improve with dataset scale. U^{AMGAE} takes the absolute value of GAE, amplifying collision and off-road penalties into high-utility signals since episodes do not terminate on collision/off-road events. This explains U^{AMGAE}'s positive correlation with collision/off-road rates in Figure 6. U^{PVL} discards negative TD errors via a ramp function, avoiding crash amplification, but its bootstrapped value targets are noisy early in training, explaining its lack of correlation with performance metrics. U^{MaxMC} uses empirical returns, avoiding both issues. The correlation between regret-based functions increases as the training set grows, because more diverse experience improves value estimation, consistent with U^{AMGAE} and U^{PVL} performing better in case 2 than in case 1.

Success-based functions target the learning frontier by construction. U^{Learn} measures Bernoulli variance $p(1-p)$, peaking at $p = 0.5$ and prioritizing scenarios where the policy succeeds half of the time. The negative correlation with return and success in Figure 6 makes sense as the dominant signal comes from later training phases, where high success ($p \to 1$) corresponds to low learnability. The positive correlation with collision/off-road rates follows the same logic: scenarios where collisions still occur have intermediate success rates and hence higher Bernoulli variance. $U^{\text{Learn-hard}}$ counts only collision-free, on-road goal completions, making it a stricter measure of frontier scenarios.

Realism-based utilities capture information orthogonal to performance. $U^{\text{GC-ADE}}$ and $U^{\text{Act-MAE}}$ are standard AD evaluation metrics repurposed as utility functions. They affect only scenario prioritization, whereas the reward function captures only goal completion and collision/off-road avoidance. Therefore, realism-based functions cannot steer behavior toward realism. This explains their lack of correlation with performance metrics in Figure 6. $U^{\text{GC-ADE}}$ and $U^{\text{Act-MAE}}$ do not correlate with each other, either, as trajectory divergence compounds over time, while action divergence does not. However, they still accelerate training relative to DR, because early in training, high behavioral divergence from logged trajectories serves as an implicit proxy for scenario difficulty. Once the policy reaches goals reliably, divergence reflects only the gap between reward-optimal and logged driving, which explains the lack of correlation across checkpoints.

Cross-category combinations are viable. The correlation analysis suggests that pairs drawn from different categories carry complementary information. We combine two utility functions by ranking scenarios independently under each and sampling from the averaged rank. Section E.5 reports three such pairs in case 1, which converge with their individual components and exceed DR throughout, and U^{MaxMC}+$U^{\text{Learn-hard}}$ sits at or above both of its components early in training. Combining a success-based function with a regret-based one therefore does not average their behavior.

5.6 How does multi-agent self-play affect curricula at scale?

Multi-agent self-play creates a persistent learning frontier that is exaggerated at scale. As discussed in Section 4, the utility of a scenario reflects the collective performance of the self-play policy. When a policy update improves some agents, it changes their trajectories, creating new conflicts for others in the same scenario. This coupling keeps scenario utility elevated, as improving some agents continuously introduces new difficulty for others. Dense scenarios amplify this as more agents means more coupled interactions. At the scale of batched simulators, each policy update consumes over half a million interactions across hundreds of scenarios, causing large policy shifts that further exaggerate the disruption and rapidly make buffer scores off-policy. We address this via higher score temperatures (see Table 4), which spread sampling probability across more scenarios.

5.7 Does reward-optimal driving imply realistic driving?

Reward optimality and behavioral realism can be orthogonal objectives. GPUDRIVE's reward function incentivizes goal completion and penalizes collisions/off-road events, but does not account for realism, e.g., comfort, speed limit compliance, or smooth lane changes. An RL agent can therefore behave optimally yet not so realistically. We run an evaluation using WOSAC metrics to assess realism (see Section E.1). DR achieves higher map-based scores than PLR variants because WOSAC measures distributional similarity to logged behavior, and uniform sampling stays closer to the data distribution. PLR variants achieve lower displacement error, consistent with more efficient goal-reaching, but diverge from human-like driving. Curricula cannot steer towards realism, as utility functions affect only scenario prioritization, not the reward. Improving both performance and realism requires reward functions that explicitly incentivize realistic or penalize unrealistic behavior.

5.8 What guidelines can we derive for practitioners?

A full-size replay buffer and a score temperature well above prior work perform best at this scale. A replay buffer equal to the training set works consistently as the dataset grows, in contrast to prior UED work, which uses buffers much smaller than the level space. Score temperatures above the typical range of $\beta \in [0.1, 1.0]$ perform better, and higher values become preferable as the dataset scales, since low temperatures concentrate sampling on top-ranked scenarios. The staleness coefficient remains in the range used in prior work [19], as higher temperature already spreads sampling across more scenarios. In addition, Section E.4 varies the sampling interval and the buffer size in case 1, where all configurations reach the same performance and exceed DR throughout, though a small buffer carries higher collision and off-road rates early in training.

On utility functions, no single choice dominates across scales, though patterns emerge. U^{MaxMC} is the most reliable regret-based function, as it does not depend on value function accuracy, whereas U^{AMGAE} and U^{PVL} underperform in case 1 and improve as the dataset grows. Success-based functions are effective at every scale and strongest in case 3, where success variance identifies the learning frontier. Realism-based functions accelerate training over DR but cannot improve realism, since they affect prioritization rather than the reward. Functions within a category prioritize similar scenarios, especially as the dataset scales, with realism as the exception. See Section B.3 for details.

6 Conclusion

In this work, we introduce CL4AD, the first integration of CL into batched AD simulators. CL4AD frames scenario selection as a UED problem, enabling adaptive prioritization of traffic scenarios via PLR [19] combined with utility functions that measure regret, success, and realism. Large-scale experiments on GPUDRIVE show that CL achieves 99% success, up to 77% faster than domain randomization, and outperforms heuristic curricula, with one exception at the largest scale, where the same heuristic offers no advantage at smaller ones. Our analysis reveals that different utility functions prioritize qualitatively different scenarios at varying rates; utility functions within the same category may correlate, but across categories capture distinct information; multi-agent self-play creates a persistent learning frontier that scale amplifies; and reward optimality and behavioral realism can be distinct axes. We derive practitioner guidelines for applying CL in batched AD simulators.

Limitations and future work. CL4AD evidences that CL scales up to the high-throughput of batched AD simulators. However, CL4AD is currently limited to variants of PLR and thus requires access to a real self-driving dataset as a source of traffic scenarios for sampling, e.g., WOMD, since GPUDRIVE operates on predefined scenarios. To address these limitations, future work will explore UED methods such as ACCEL [29], which randomly mutates prioritized scenarios, hence increasing scenario diversity for training robust policies. Applying ACCEL to GPUDrive requires a mutation operator over scenarios, e.g., one that perturbs the initial positions and velocities of agents, inserts or removes agents, or edits their goals, while keeping the scene drivable and consistent with the road graph. PAIRED and RE-PAIRED instead require a teacher policy that emits such scenario parameters directly, which entails a generative scenario representation rather than the identification numbers GPUDrive currently operates with. In addition, synthetic scenario generation tools, e.g., Scenario Dreamer [32], can enable CL4AD to further accelerate training and improve the robustness and generalization capabilities of trained agents by creating safety-critical or out-of-distribution scenarios that the agent struggles with. CL4AD also trains on all sampled scenarios, whereas Robust PLR suppresses gradient updates on newly sampled levels and reports improved zero-shot transfer under sharper prioritization than our score temperatures induce. Whether that restriction transfers to batched self-play remains untested.

References

[1] Ahmed Abouelazm, Tim Weinstein, Tim Joseph, Philip Schörner, and J Marius Zöllner. Automatic curriculum learning for driving scenarios: Towards robust and efficient reinforcement learning. *arXiv preprint arXiv:2505.08264*, 2025.

[2] Luca Anzalone, Silvio Barra, and Michele Nappi. Reinforced curriculum learning for autonomous driving in carla. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3318–3322. IEEE, 2021.

[3] Luca Anzalone, Paola Barra, Silvio Barra, Aniello Castiglione, and Michele Nappi. An end-to-end curriculum learning approach for autonomous driving scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):19817–19826, 2022.

[4] Adrien Baranes and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration for active motor learning in robots: A case study. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1766–1773. IEEE, 2010.

[5] Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, et al. Human-timescale adaptation in an open-ended task space. In *International Conference on Machine Learning*, pages 1887–1935. PMLR, 2023.

[6] Axel Brunnbauer, Luigi Berducci, Peter Priller, Dejan Nickovic, and Radu Grosu. Scenario-based curriculum generation for multi-agent autonomous driving. *arXiv preprint arXiv:2403.17805*, 2024.

[7] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

[8] Daphne Cornelisse and Eugene Vinitsky. Human-compatible driving agents through data-regularized self-play reinforcement learning. In *Reinforcement Learning Conference*, 2024.

[9] Daphne Cornelisse, Aarav Pandya, Kevin Joseph, Joseph Suárez, and Eugene Vinitsky. Building reliable sim driving agents by scaling self-play. *arXiv preprint arXiv:2502.14706*, 2025.

[10] Marco Francis Cusumano-Towner, David Hafner, Alexander Hertzberg, Brody Huval, Aleksei Petrenko, Eugene Vinitsky, Erik Wijmans, Taylor W. Killian, Stuart Bowers, Ozan Sener, Philipp Kraehenbuehl, and Vladlen Koltun. Robust autonomy emerges from self-play. In *Forty-second International Conference on Machine Learning*, 2025.

[11] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

[12] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

[13] 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.

[14] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9710–9719, 2021.

[15] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[16] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36:7730–7742, 2023.

[17] Matthew T Jackson, Minqi Jiang, Jack Parker-Holder, Risto Vuorio, Chris Lu, Greg Farquhar, Shimon Whiteson, and Jakob Foerster. Discovering general reinforcement learning algorithms with adversarial environment design. *Advances in Neural Information Processing Systems*, 36: 79980–79998, 2023.

[18] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021.

[19] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021.

[20] Minqi Jiang, Michael D Dennis, Edward Grefenstette, and Tim Rocktäschel. minimax: Efficient baselines for autocurricula in JAX. In *Second Agent Learning in Open-Endedness Workshop*, 2023.

[21] Ingmar Kanitscheider, Joost Huizinga, David Farhi, William Hebgen Guss, Brandon Houghton, Raul Sampedro, Peter Zhokhov, Bowen Baker, Adrien Ecoffet, Jie Tang, et al. Multi-task curriculum learning in a complex, visual, hard-exploration domain: Minecraft. *arXiv preprint arXiv:2106.14876*, 2021.

[22] Saman Kazemkhani, Aarav Pandya, Daphne Cornelisse, Brennan Shacklett, and Eugene Vinitsky. GPUDrive: Data-driven, multi-agent driving simulation at 1 million FPS. In *The Thirteenth International Conference on Learning Representations*, 2025.

[23] Pascal Klink, Haoyi Yang, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Curriculum reinforcement learning via constrained optimal transport. In *International Conference on Machine Learning*, pages 11341–11358. PMLR, 2022.

[24] Cevahir Koprulu, Thiago D Simão, Nils Jansen, and Ufuk Topcu. Risk-aware curriculum generation for heavy-tailed task distributions. In *Uncertainty in Artificial Intelligence*, pages 1132–1142. PMLR, 2023.

[25] Quanyi Li, Zhenghao Mark Peng, Lan Feng, Zhizheng Liu, Chenda Duan, Wenjie Mo, and Bolei Zhou. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. *Advances in neural information processing systems*, 36:3894–3920, 2023.

[26] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nicholas Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, et al. The waymo open sim agents challenge. *Advances in Neural Information Processing Systems*, 36:59151–59171, 2023.

[27] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[28] Haoyi Niu, Yizhou Xu, Xingjian Jiang, and Jianming Hu. Continual driving policy optimization with closed-loop individualized curricula. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6850–6857. IEEE, 2024.

[29] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pages 17473–17498. PMLR, 2022.

[30] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853. PMLR, 2020.

[31] Zhiqian Qiao, Katharina Muelling, John M Dolan, Praveen Palanisamy, and Priyantha Mudalige. Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1233–1238. IEEE, 2018.

[32] Luke Rowe, Roger Girgis, Anthony Gosselin, Liam Paull, Christopher Pal, and Felix Heide. Scenario dreamer: Vectorized latent diffusion for generating driving simulation environments. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17207–17218, 2025.

[33] Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. *Advances in Neural Information Processing Systems*, 37:16071–16101, 2024.

[34] Erdi Sayar, Giovanni Iacca, Ozgur S Oguz, and Alois Knoll. Diffusion-based curriculum reinforcement learning. *Advances in Neural Information Processing Systems*, 37:97587–97617, 2024.

[35] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

[37] Zihao Sheng, Zilin Huang, Yansong Qu, Yue Leng, Sruthi Bhavanam, and Sikai Chen. Curricu-vlm: Towards safe autonomous driving via personalized safety-critical curriculum learning with vision-language models. *Transportation Research Part C: Emerging Technologies*, 185:105549, 2026.

[38] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[39] Ryan Sullivan, Ryan Pégoud, Ameen Ur Rehman, Xinchen Yang, Junyun Huang, Aayush Verma, Nistha Mitra, and John P Dickerson. Syllabus: Portable curricula for reinforcement learning agents. *Reinforcement Learning Journal*, 6:1816–1855, 2025.

[40] Georgios Tzannetos, Bárbara Gomes Ribeiro, Parameswaran Kamalaruban, and Adish Singla. Proximal curriculum for reinforcement learning agents. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

[41] Eugene Vinitsky, Nathan Lichtlé, Xiaomeng Yang, Brandon Amos, and Jakob Foerster. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *Advances in Neural Information Processing Systems*, 35:3962–3974, 2022.

[42] Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. OMNI: Open-endedness via models of human notions of interestingness. In *The Twelfth International Conference on Learning Representations*, 2024.

[43] Linrui Zhang, Zhenghao Peng, Quanyi Li, and Bolei Zhou. Cat: Closed-loop adversarial training for safe end-to-end driving. In *Conference on Robot Learning*, pages 2357–2372. PMLR, 2023.

A Nomenclature

$\mathcal{G}$	POSG
$\mathcal{N}$, N	Set of agents, number of agents ($\lvert\mathcal{N}\rvert = \text{N}$) in POSG
$\mathcal{S}$, $\mathcal{A}$, $\mathcal{O}$	State, action and observation spaces in POSG
$\mathbf{s}$, $\mathbf{a}$, $\mathbf{o}$, r	State, action, observation, and reward in POSG
T, Z, R	Transition, observation, and reward functions in POSG
I	Initial state distribution in POSG
γ, H	Discount factor and horizon in POSG
$\mathcal{G}^\Theta$, Θ, θ	UPOSG, its set of scenarios and a scenario, i.e., $\theta \in \Theta$
$\mathcal{N}^\Theta$, N	Set of agents, number of agents ($\lvert\mathcal{N}^\Theta\rvert = \text{N}$) in UPOSG
$\mathcal{S}$, $\mathcal{A}^\Theta$, $\mathcal{O}^\Theta$	State, action and observation spaces in UPOSG
$\mathbf{s}$, $\mathbf{a}$, $\mathbf{o}$, r	State, action, observation, and reward in UPOSG
T^Θ, Z^Θ, R^Θ, I^Θ	Transition, observation, reward functions and initial state distribution in UPOSG
γ, H	Discount factor and horizon in UPOSG
$\mathbf{x}$	Position of an agent in a scenario in UPOSG
M	Number of scenarios in a UPOSG, i.e., $\lvert\Theta\rvert = \text{M}$
$\mathcal{S}^\theta_{\text{Goal}}$	Goal states in a UPOSG
Λ	Level generator for UED
Π	Policy space in UED
$\Delta(\Theta)$	Distribution over levels in UED
U, C	Utility function in UED, constant utility in UED
$\mathcal{B}$, $\mathbb{P}_{\text{replay}}$	Replay buffer and distribution in PLR
$\mathbb{P}_{\text{utility}}$, $\mathbb{P}_{\text{staleness}}$	Score and staleness distribution in PLR
l	Scenario sampling iteration in PLR
ρ, β, d, B^{max}	Staleness coefficient, score temperature, replay rate, max replay buffer size
π, V	Policy, value function
λ, δ	GAE discount factor, TD error
$\text{R}^{\theta,n}_{\text{max}}$	Maximum return of an agent
p	Success rate
ϕ	Trainable policy parameter
T^{train}, T^{sce}, T^{pol}	Number of interactions for training, sampling scenarios, and updating policy
W	Number of concurrent worlds
$\mathcal{D}$	Experience buffer
$\Phi()$	RL algorithm of choice to update policy
e	End of episode flag
τ	Rollout

Algorithm 2 SAMPLEFROMCURRICULUM()

Input: Replay buffer $\mathcal{B}$, set of training scenarios Θ^{train}, sampling iteration l
Parameters: Replay rate d, staleness ρ, temperature β, max buffer size B^{max}, number of worlds W
Output: Sampled scenarios $(\theta_w)_{w=1}^{\text{W}}$, and buffer $\mathcal{B}$ with updated staleness

1: $\mathcal{B} \leftarrow \text{DISCARDLOWESTRANKINGSCENARIOS}(\mathcal{B}, \text{B}^{\text{max}})$
2: **if** $|\mathcal{B}| \equiv 0$ **or** $(\text{Bernoulli}(d) \equiv 0$ **and** $|\Theta^{\text{train}} - \mathcal{B}^{\text{scenario}}| > 0)$ **then**
3: $\quad \mathbb{P}_{\text{sample}} \leftarrow \text{Uniform}(\Theta^{\text{train}} - \mathcal{B}^{\text{scenario}})$ ▷ *Uniformly randomly sample scenarios*
4: **else**
5: $\quad \mathbb{P}_{\text{sample}} \leftarrow \mathbb{P}_{\text{replay}}$ ▷ *Replay scenarios based on* $\mathbb{P}_{replay}$
6: **end if**
7: $(\theta_w)_{w=1}^{\text{W}} \leftarrow \text{Sample}(\mathbb{P}_{\text{sample}}, \text{W})$ ▷ *Sample W-many scenarios based on* $\mathbb{P}_{sample}$
8: $\mathcal{B}^{\text{scenario}} \leftarrow \mathcal{B}^{\text{scenario}} \cup (\theta_w)_{w=1}^{\text{W}}$ ▷ *Update scenarios in the replay buffer*
9: $l_{\theta_w} \leftarrow l, \forall w \in [W]$ ▷ *Update sampling iteration for staleness distribution*
10: $\tau_{\theta_w} \leftarrow (), \forall w \in [W]$ ▷ *Reset the rollout*

Algorithm 3 UPDATECURRICULUM()

Input: Interaction set $\mathcal{D}_t$, utility function U, replay buffer $\mathcal{B}$
Output: Updated replay buffer $\mathcal{B}$

1: **for** $w \in [\text{W}]$ **do**
2: $\quad$ **if** $e_{n,w}$ is True $\forall n \in [\text{N}_{\theta_w}]$ **then**
3: $\quad\quad \text{score}_{\theta_w,t} \leftarrow U(\tau_{\theta_w})$ ▷ *Compute utility score for terminated episode*
4: $\quad\quad \text{score}_{\theta_w} \leftarrow \text{MovingAverage}(\text{score}_{\theta_w}, \text{score}_{\theta_w,t})$ ▷ *Update the score in the buffer*
5: $\quad\quad \tau_{\theta_w} \leftarrow ()$ ▷ *Reset the rollout*
6: $\quad$ **else**
7: $\quad\quad \tau_{\theta_w} \leftarrow \tau_{\theta_w} \cup \{\mathbf{o}_{n,w}, \mathbf{a}_{n,w}, \mathbf{o}'_{n,w}, r_{n,w}, e_{n,w}\}_{n \in [\text{N}_{\theta_w}]}$ ▷ *Update the rollout with new interactions*
8: $\quad$ **end if**
9: **end for**

B Details of CL4AD

In this section, we provide a more detailed look into how CL4AD works to support the material in Section 4 and share the wall-clock overhead of curriculum updates and sampling.

B.1 Curriculum Sampling in CL4AD

The score distribution $\mathbb{P}_{\text{utility}}(\theta_i|\mathcal{B}, U)$ is based on the ranking of seen levels,

$$\mathbb{P}_{\text{utility}}(\theta_i|\mathcal{B}, U) = \frac{\text{rank}(\theta_i|\mathcal{B})^{-1/\beta}}{\sum_{j \in \mathcal{B}^{\text{scenario}}} \text{rank}(\theta_j|\mathcal{B})^{-1/\beta}}, \tag{2}$$

with a temperature parameter β tuning the impact of ranking. The staleness distribution assigns a higher likelihood for levels that has not been sampled for longer, namely,

$$\mathbb{P}_{\text{staleness}}(\theta_i|\mathcal{B}, l) = \frac{l - l_{\theta_i}}{\sum_{j \in \mathcal{B}^{\text{scenario}}} l - l_{\theta_j}}, \tag{3}$$

where l is the total number of sampling iterations so far, and l_{θ_j} is the iteration at which scenario θ_j was last sampled. Algorithm 2 is a pseudocode for how CL4AD samples new scenarios during training via PLR. First, CL4AD removes scenarios with ranking lower than B^{max} in the buffer (Line 1), where B^{max} is the maximum size of $\mathcal{B}$ for sampling. If the buffer size is smaller than or equal to B^{max}, then no scenario is removed. Then, it determines whether to sample traffic scenarios from the replay buffer. If the replay buffer is empty, or the random replay decision is False, conditioned on the fact that there are still unseen scenarios, then CL4AD uniformly randomly samples unseen scenarios from the training dataset. Otherwise, it uses the replay distribution $\mathbb{P}_{\text{replay}}$ to sample from the replay buffer $\mathcal{B}$ (lines 2-6). Then, CL4AD updates the scenarios in the buffer with the newly sampled ones and sets their corresponding last sampling iteration to the current one for staleness computation later on (lines 7-9).

B.2 Curriculum Updates in CL4AD

Algorithm 3 is a pseudocode for how CL4AD updates the buffer. CL4AD goes through every world and checks whether an episode has terminated. If so, it computes the utility of that episode based on the rollout that CL4AD has kept track of. Then it stores this score, and at the next sampling call, it averages the scores of all episodes in that scenario to update the buffer. Finally, the rollout is reset for a new episode to save memory. If the episode continues, CL4AD updates the rollouts with the latest interactions.

B.3 Practitioner Guidelines

Based on our experimental findings, we offer the following recommendations to practitioners.

PLR configuration at scale.

1) **Set the replay buffer size equal to the training dataset size.** Prior UED work uses buffers much smaller than the level space; our experiments show that a full-size buffer works consistently as the dataset scales.
2) **Use a score temperature significantly higher than prior work.** We use $\beta \in \{2, 4\}$, well above the typical range of $\beta \in [0.1, 1.0]$. Lower temperatures concentrate sampling on a few top-ranked scenarios, which becomes problematic as the buffer grows. Increasing β generally leads to better performance as the dataset scales up. Higher temperature also indirectly mitigates staleness by spreading sampling across more scenarios.
3) **The staleness coefficient can remain in the same range as prior work,** $\rho \in \{0.1, 0.3\}$ [19], as higher score temperature already alleviates staleness.

Utility function selection.

1) **No single utility function dominates across all scales.** However, several patterns emerge from our results (see Sections 5.2 and 5.5).
2) **Among regret-based functions, MaxMC is the most reliable across scales,** as it does not depend on value function accuracy. AMGAE and PVL rely on value estimation and can underperform at smaller scale (case 1). As the dataset scales, their performance improves (case 2), consistent with the increasing correlation between regret-based functions in Figure 6.
3) **Success-based functions are effective across all scales** and particularly strong at large scale (case 3), where identifying the learning frontier through success variance is effective.
4) **Realism-based functions accelerate training over DR but cannot improve realism,** as they only affect scenario prioritization, not the reward function.
5) **Utility functions within the same category tend to prioritize similar scenarios,** especially as the dataset scales, except the realism category (see Figure 6).

B.4 Computational overhead of CL4AD

To quantify the overhead of curriculum learning, we measured wall-clock times for all CL4AD components during training. Curriculum updates (utility computation and buffer score updates) and scenario sampling (PLR sampling and scenario assignment to worlds) occur every $\mathrm{T}^{\mathrm{sce}} = 2,000,000$ interactions.

Per inter-sampling segment, curriculum update time totals $\sim 4.3s$, which is $\sim 1\%$ of the segment's evaluation/rollout time ($\sim 428.5s$). The PLR sampling operation itself takes $\sim 0.66ms$ per call, the remaining $\sim 6.15s$ of each sampling block is spent assigning scenarios to worlds and resetting the simulator, which is simulator overhead, not CL4AD overhead.

Over a full training run of one billion steps ($\sim$ 110 hours on an A5000, Section D), there are approximately 500 curriculum sampling steps. The total curriculum update time is approximately 36 minutes, and the total PLR sampling time is approximately 0.3 seconds. The overhead of assignment of scenarios ($\sim$ 51 minutes total) is attributed to the simulator, not to CL4AD. In total, CL4AD adds less than 1% to the training wall-clock time.

Table 1: Wall-clock time per inter-sampling segment

Component	Time	Percentage
Curriculum update (total)	4.30 ± 0.21s	1.04%
Evaluation/rollout time	428.50 ± 88.44s	100%

Table 2: CL4AD sampling vs. scenario assignment per curriculum sampling call

Component	Time	Percentage
Curriculum sampling	0.66 ± 0.02ms	0.01%
Scenario assignment to worlds	6.15 ± 0.40s	99.99%

C Experimental Details

In this section, we describe the process of hyperparameter selection for our experiments.

C.1 Simulation Set-up

Our integration of CL4AD into GPUDRIVE follows the simulation set-up in Kazemkhani et al. [22], where the simulator ignores collisions and going off-road, i.e., they do not lead to episode termination; the observation of a vehicle is its bird-eye-view of a radius of 50m; non-vehicle objects are omitted; a goal is considered to be achieved if an agent is in its proximity by 2m; the action consists of two discrete random variables for steering and acceleration inputs, divided into evenly spaced grids, 13 and 7, respectively; maximum number of controlled agents in a scenario is 64; the agents only observe the current time step; and the episode takes 91 timesteps, amounting to 9 seconds, with 1 second of history followed by 8 seconds of future. The public test partition withholds the future trajectories, as they are the prediction targets of the motion forecasting benchmark, leaving 11 logged steps per scenario. Since GPUDrive sets each agent's goal to its final logged position, goals in test scenarios lie roughly one second of driving ahead, and agents are exposed to fewer timesteps in which collisions and off-road events can occur. For more details, we refer the reader to the default PufferLib configuration (see `environment` section) in the repository published by Kazemkhani et al. [22].

C.2 Self-play PPO Training

Self-play RL is an RL scheme for multi-agent settings where each agent samples their actions from a shared, decentralized policy. More formally, this scheme samples the action $\mathbf{a}_{i,t} \sim \pi_\phi(\mathbf{o}_{i,t})$ of agent $i \in \mathcal{N}^\Theta$ via a policy π_ϕ parameterized by ϕ, e.g., a neural network with learnable parameters ϕ, given the observation $\mathbf{o}_{i,t}$ of said agent at time t. Batched AD simulators GIGAFLOW and GPUDRIVE use self-play RL as the strategy to train a single policy that controls all vehicles in a scenario in parallel. Their batched structure empowers parallelization further by concurrently simulating hundreds to thousands of traffic scenarios to accelerate experience collection. Both works employ an on-policy RL algorithm, proximal policy optimization (PPO) [36], where policy updates occur once the simultaneous data collection fills an experience buffer. As a result, batched simulation accelerates experience collection via parallelized scenarios, while self-play RL saves compute time and memory by training a single policy. We implement CL4AD on GPUDRIVE, which samples hundreds of traffic scenarios every couple of million interactions, with initial positions and goals from logged data in WOMD. The default scenario sampling is uniform, where each traffic scenario has equal likelihood.

Table 3 lists the hyperparameters for self-play PPO training in cases 1, 2, and 3, as well as the ablation study. As the ablation study investigates limited compute resources, i.e., the use of fewer worlds and lower batch sizes, we essentially set them according to the hyperparameters in Kazemkhani et al. [22], where the number of worlds $\mathrm{W} = 50$. In comparison, cases 1, 2, and 3 studies a larger scale in terms of throughput, hence utilize significantly more concurrent worlds and a larger experience buffer. As a result, their hyperparameters come from Cornelisse et al. [9], which focuses on a similar scale. The weights for collision/off-road penalties and goal completion rewards also come from Cornelisse et al. [9]. The experiments are over three independent runs, utilizing seeds 42, 12, and 67. The network architecture also follows the settings in Cornelisse et al. [9].

Table 3: Self-play PPO Hyperparameters

Parameter	Case 1,2,3	Ablation
`total_timesteps` T^{train}	$2,000,000,000$	$1,000,000,000$
`num_worlds` W	800	100
`batch_size` T^{pol}	$524,288$	$131,072$
`minibatch_size`	$16,384$	$8,192$
`learning_rate`	0.0003	0.0003
`anneal_lr`	`false`	`false`
`gamma` γ	0.99	0.99
`gae_gamma` λ	0.95	0.95
`update_epochs`	2	4
`norm_adv`	`true`	`true`
`clip_coef`	0.2	0.2
`clip_vloss`	`false`	`false`
`vf_clip_coef`	0.2	0.2
`ent_coef`	0.0001	0.0001
`vf_coef`	0.5	0.3
`max_grad_norm`	0.5	0.5
`target_kl`	`null`	`null`
`collision_weight`	-0.75	-0.75
`off_road_weight`	-0.75	-0.75
`goal_achieved_weight`	1.0	1.0

C.3 Scenario Sampling Details

Table 4 demonstrates the hyperparameters used for the experiments we report in cases 1, 2, 3, and the ablation study. The search space for PLR hyperparameters is as follows: staleness coefficient $\rho \in \{0.1, 0.3\}$ and score temperature $\beta \in \{2, 4\}$, based mainly on Jiang et al. [19]. We first conduct a grid search in Case 1, where we train agents using all score functions on three independent runs for one billion interactions. Then we select the pair that yields the highest success rate, the fastest at test-time. Jiang et al. [18] suggests a lower temperature; however, our experiments indicate that a higher temperature, especially considering the size of the training dataset, is more performant in large-scale training. Case 3 and the ablation study also utilize these hyperparameters. In case 2, we find that a higher temperature yields better results. We set the replay buffer size to the size of the training dataset, and sample scenarios every $2,000,000$ interactions. For heuristic curricula, once scenarios are ranked using the chosen heuristic, we sample from a score-rank distribution $\mathbb{P}_{\text{utility}}$ with fixed utility and $\beta \in \{2, 4\}$.

D Computational Resources

We run our experiments in cases 1, 2, and 3 on an NVIDIA H200, which has 141 GB of GPU memory. One training run, which amounts to 2 billion steps and approximately 3,800 policy updates, takes around 60 hours. For the ablation study, we train agents on NVIDIA RTX A5000, which has a GPU memory of 24GB, for a billion interactions, which takes over 110 hours.

E Detailed Results

E.1 WOSAC Evaluation

We evaluate the final trained policies using the Waymo Open Sim Agents Challenge (WOSAC) metrics [26] in two settings: (1) all controlled agents via self-play, and (2) ego-only with other agents replaying logged trajectories. Table 5 and Table 6 report the results for case 1 (150 test scenarios, 3 seeds).

Kinematics and interaction scores are comparable across all methods, indicating that the curriculum choice does not meaningfully affect these categories. The main difference is in map-based metrics (Dist Edge, Offroad), where DR scores higher because WOSAC measures distributional similarity to

Table 4: PLR Hyperparameters

	Utility Function	d	β	ρ
Case 1	$U^{\text{Act-MAE}}$	0.5	2	0.3
	U^{AMGAE}	0.5	4	0.3
	$U^{\text{GC-ADE}}$	0.5	4	0.3
	U^{Learn}	0.5	4	0.1
	$U^{\text{Learn-hard}}$	0.5	2	0.1
	U^{MaxMC}	0.5	2	0.3
	U^{PVL}	0.5	4	0.3
Case 2	$U^{\text{Act-MAE}}$	0.5	4	0.3
	U^{Learn}	0.5	4	0.1
	U^{MaxMC}	0.5	4	0.3
Case 3	$U^{\text{Act-MAE}}$	0.5	4	0.3
	U^{Learn}	0.5	4	0.1
	U^{MaxMC}	0.5	2	0.3
Ablation	$U^{\text{Act-MAE}}$	0.5	2	0.3
	U^{Learn}	0.5	4	0.1
	U^{MaxMC}	0.5	2	0.3

Table 5: WOSAC metrics, self-play evaluation with final trained policies (case 1: 150 test scenarios).

Method	Realism ↑	minADE ↓	Lin Spd ↑	Lin Acc ↑	Ang Spd ↑	Ang Acc ↑	Dist Obj ↑	Collis ↑	TTC ↑	Dist Edge ↑	Offroad ↑
Oracle	0.832	0.00	0.493	0.448	0.578	0.694	0.430	0.999	0.900	0.772	0.999
DR	0.689	10.28	0.161	0.246	0.510	0.658	0.163	0.855	0.856	0.703	0.901
PLR+MaxMC	0.657	9.20	0.165	0.252	0.514	0.662	0.158	0.830	0.857	0.701	0.804
PLR+GC-ADE	0.652	9.58	0.158	0.229	0.509	0.660	0.152	0.840	0.856	0.691	0.783
PLR+Learn-Hard	0.649	9.76	0.158	0.230	0.508	0.660	0.150	0.837	0.856	0.694	0.778
PLR+AMGAE	0.644	9.94	0.159	0.244	0.508	0.659	0.156	0.849	0.854	0.674	0.747
PLR+PVL	0.640	9.71	0.157	0.231	0.510	0.660	0.152	0.858	0.857	0.670	0.724
PLR+Act-MAE	0.631	9.40	0.158	0.212	0.512	0.664	0.147	0.824	0.856	0.689	0.724
PLR+Learn	0.628	10.10	0.153	0.217	0.507	0.663	0.146	0.863	0.856	0.648	0.685

logged behavior, and uniform sampling stays closer to the data distribution. PLR variants achieve lower displacement error, consistent with more efficient goal reaching, but diverge from human driving patterns in map-based metrics. This gap widens in the ego-only setting. All methods remain far from the oracle, with kinematics representing the largest gap. These results support the analysis in Section 5.7: the curriculum improves task performance but cannot improve realism without changes to the reward function.

E.2 Quantitative Results

Figures 7, 8, 9, and 10 demonstrate the progression of trained agents in cases 1, 2, and 3, as well as the compute ablation, respectively. These figures provide details on the progression of performance, regret, realism, and learnability when agents are evaluated in the training and test partitions of their respective experiments. Regret, learnability, and realism in the training partition highlight how automated curricula impact training. In most cases, we observe that PLR variants are significantly faster than DR at achieving low utility scores in these metrics, indicating that they obtain more performant and realistic policies more quickly. The performance progression, when evaluated on the training partition, leads to a similar observation as well. Progression in test scenarios demonstrates the generalization capabilities of these trained agents, as these scenarios were not encountered during training. Overall, we observe that PLR variants are again quickly becoming more capable at generalization or becoming robust and reliable faster than agents trained via DR.

Table 6: WOSAC metrics, ego-only evaluation with final trained policies (case 1: 150 test scenarios).

Method	Realism ↑	minADE ↓	Lin Spd ↑	Lin Acc ↑	Ang Spd ↑	Ang Acc ↑	Dist Obj ↑	Collis ↑	TTC ↑	Dist Edge ↑	Offroad ↑
Oracle	0.871	0.00	0.582	0.694	0.721	0.917	0.482	0.999	0.918	0.854	0.999
DR	0.669	9.14	0.279	0.402	0.617	0.864	0.144	0.769	0.843	0.724	0.804
PLR+MaxMC	0.624	8.25	0.277	0.405	0.620	0.862	0.136	0.765	0.842	0.674	0.649
PLR+PVL	0.608	8.47	0.274	0.386	0.615	0.866	0.125	0.766	0.844	0.659	0.597
PLR+GC-ADE	0.603	8.58	0.273	0.385	0.612	0.863	0.132	0.753	0.842	0.661	0.591
PLR+AMGAE	0.595	8.72	0.273	0.388	0.612	0.864	0.128	0.749	0.840	0.637	0.572
PLR+Learn-Hard	0.586	8.45	0.274	0.385	0.613	0.868	0.126	0.757	0.842	0.636	0.529
PLR+Act-MAE	0.572	8.35	0.271	0.369	0.615	0.864	0.124	0.761	0.843	0.625	0.478
PLR+Learn	0.556	9.00	0.267	0.373	0.611	0.861	0.120	0.768	0.841	0.565	0.424

E.3 Qualitative Results

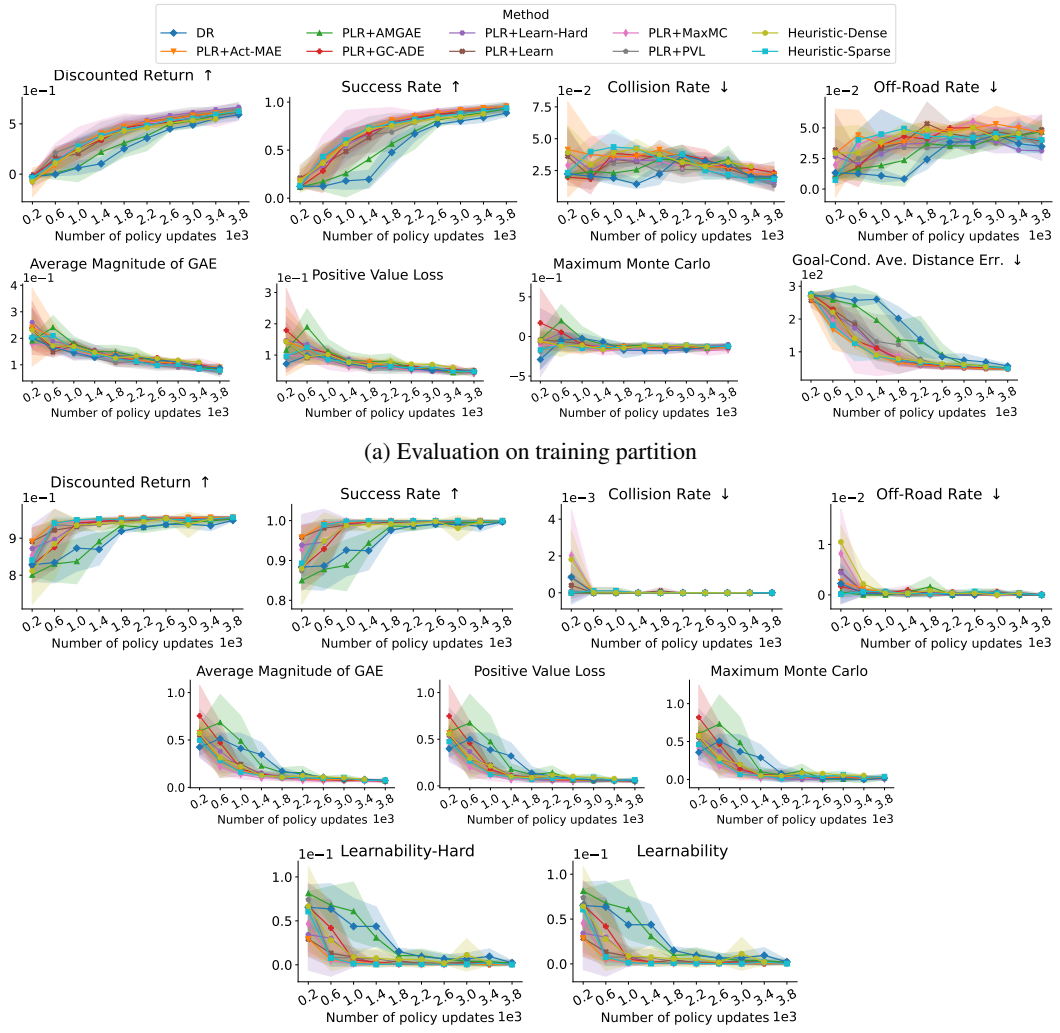
Figures 11, 12, 13, 14, 15, and 16 illustrate the $\mathbb{P}_{\text{replay}}$ progression of PLR in case 1. Here we omit U^{MaxMC}, as we provide its illustration in the main document. The utility functions with a high score temperature, i.e., $\beta = 4$, as opposed to $\beta = 2$, lead to a more uniform replay distribution (see Figures 12, 13, 15, 16 for U^{AMGAE}, $U^{\text{GC-ADE}}$, U^{Learn}, and U^{PVL}, respectively). As the score temperature decreases, the impact of the ranking on the replay distribution also decreases. Furthermore, we observe that certain utility functions result in significant changes in the replay distribution throughout training, specifically when visualized with respect to the number of controlled agents in scenarios (see Figures 11 and 14 for $U^{\text{Act-MAE}}$ and $U^{\text{Learn-hard}}$, respectively). This change may be due to a lower score temperature, which allows the ranking to impact the replay distribution more strongly.

E.4 Sensitivity to sampling interval and buffer size

We ablate the two PLR hyperparameters that govern how often scores are refreshed and how many scenarios the buffer retains. Our reported configuration uses a buffer equal to the training set and $\text{T}^{\text{sce}} = 2 \times 10^6$ interactions, which is the scenario sampling interval GPUDrive uses under domain randomization. We compare against three variations, all with U^{MaxMC} in case 1 across three seeds: **(1)** Small Buffer: a buffer of 100 scenarios, i.e., $|\mathcal{B} = 100|$, and $\text{T}^{\text{sce}} = 2 \times 10^6$, **(2)** Faster Sampling: full buffer, i.e., $|\mathcal{B}| = 1,000$, and $\text{T}^{\text{sce}} = 1 \times 10^6$, and **(3)** Slower Sampling: full buffer, i.e., $|\mathcal{B}| = 1,000$, and $\text{T}^{\text{sce}} = 4 \times 10^6$. Figure 17 reports the progression on the test split. All four configurations reach the same discounted return and success rate by 1,400 policy updates, with collision and off-road rates near zero from 1,000 updates onward, and all four exceed DR in return and success rate at every checkpoint, so CL4AD is insensitive to both hyperparameters over the ranges we test. The configurations differ only over the first few hundred updates, where faster sampling and the small buffer reach high success earliest and slower sampling trails the other three. This follows from the staleness mechanism in Section 5.6: shorter sampling intervals keep buffer scores closer to on-policy, and a small buffer has a similar effect, as evicting all but the top-ranked 10% of scenarios causes the retained ones to be revisited more often and prioritized more sharply. The small buffer carries elevated collision and off-road rates through 600 updates, since the curriculum spends less of its budget on the broader set of scenarios where the remaining safety failures occur.

E.5 Combining utility functions across categories

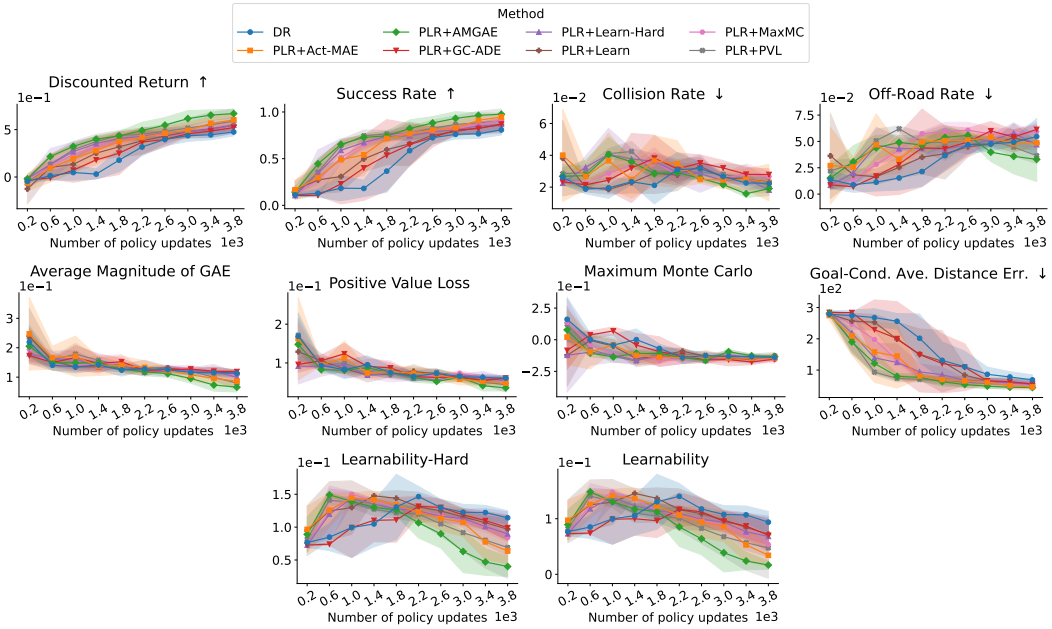
Section 5.5 shows that utility functions correlate within a category, except for realism, but not across categories, suggesting that cross-category pairs carry complementary information. We combine two utility functions by ranking the scenarios independently under each and sampling from the averaged rank, preserving the scale invariance of $\mathbb{P}_{\text{utility}}$ in Equation (2), since a weighted sum of raw scores does not consist of utilities expressed in different units. We evaluate three pairs in case 1 across three seeds, one from each pairwise combination of categories: **(1)** U^{MaxMC} + $U^{\text{Learn-hard}}$, **(2)** U^{MaxMC} + $U^{\text{Act-MAE}}$, and **(3)** $U^{\text{Learn-hard}}$ + $U^{\text{Act-MAE}}$. Figure 18 reports the progression on the test split. All three combinations converge with their individual components in discounted return and success rate by 1,400 policy updates, with collision and off-road rates near zero, and all exceed DR at every checkpoint. $U^{\text{Learn-hard}}$ alone is the slowest PLR variant in this setting, whereas U^{MaxMC} + $U^{\text{Learn-hard}}$ sits at or above both components at the early checkpoints, so combining a success-based function with a regret-based one does not average their behavior. One explanation, which we do not test here, is that the combined ranking draws from both density regimes, since Section 5.2 shows regret-based functions drifting above the dataset average traffic density while success-based functions shift below.



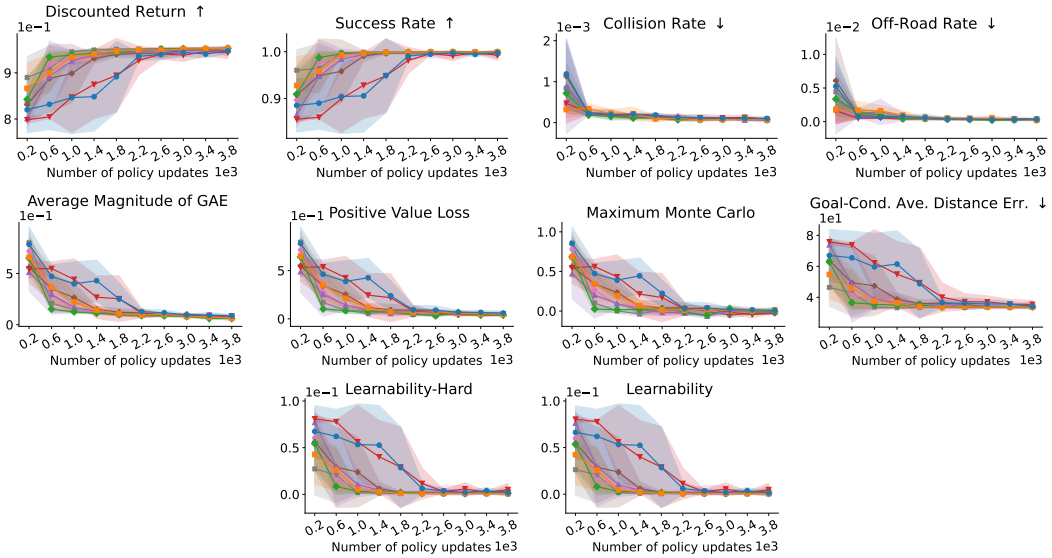
(a) Evaluation on training partition

(b) Evaluation on test partition

Figure 7: Case 1: Regret (U^{AMGAE}, U^{PVL}, U^{MaxMC}), realism ($U^{\text{GC-ADE}}$), and learnability (U^{Learn}, $U^{\text{Learn-hard}}$), progression during training with 1000 scenarios from WOMD: We evaluate in **(a)** training partition, and **(b)** 150 test scenarios. Bold markers indicate the mean, whereas the shaded area covers one standard deviation around it across three independent training runs.



(a) Evaluation on training partition



(b) Evaluation on test partition

Figure 8: Case 2: Performance, Regret (U^{AMGAE}, U^{PVL}, U^{MaxMC}), realism ($U^{\text{GC-ADE}}$), and learnability (U^{Learn}, $U^{\text{Learn-hard}}$), progression during training with 10,000 scenarios from WOMD: We evaluate in **(a)** training partition, and **(b)** 10,000 test scenarios. Bold markers indicate the mean, whereas the shaded area covers one standard deviation around it across three training runs.

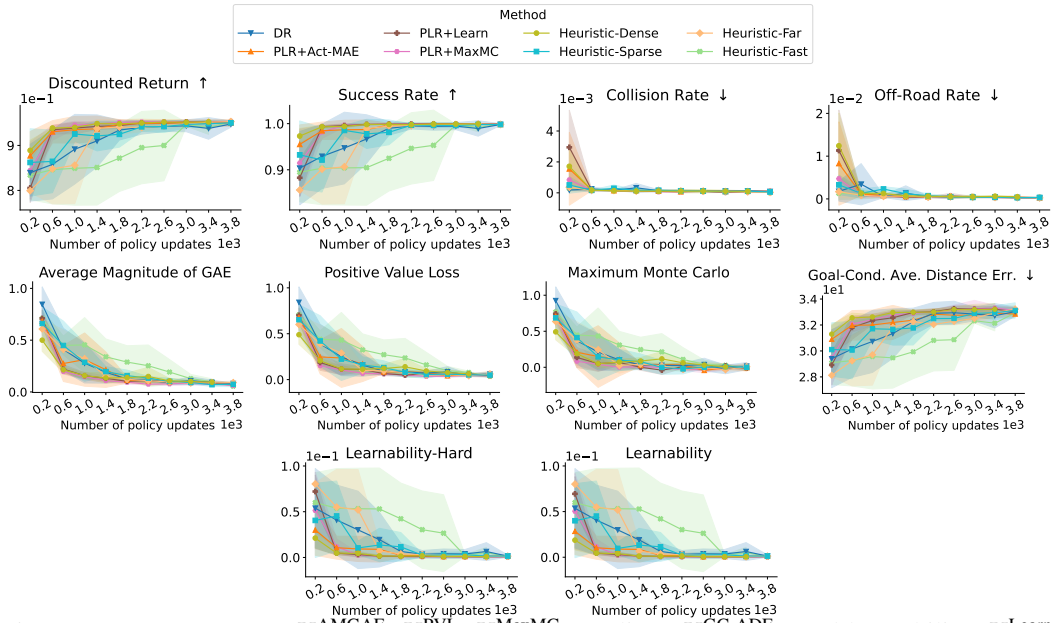
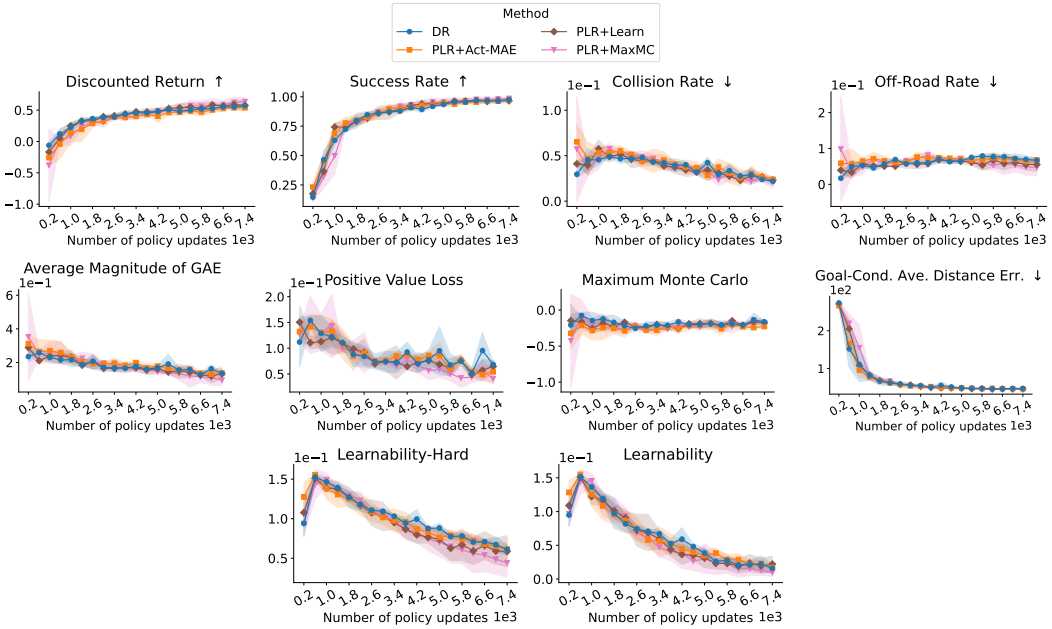
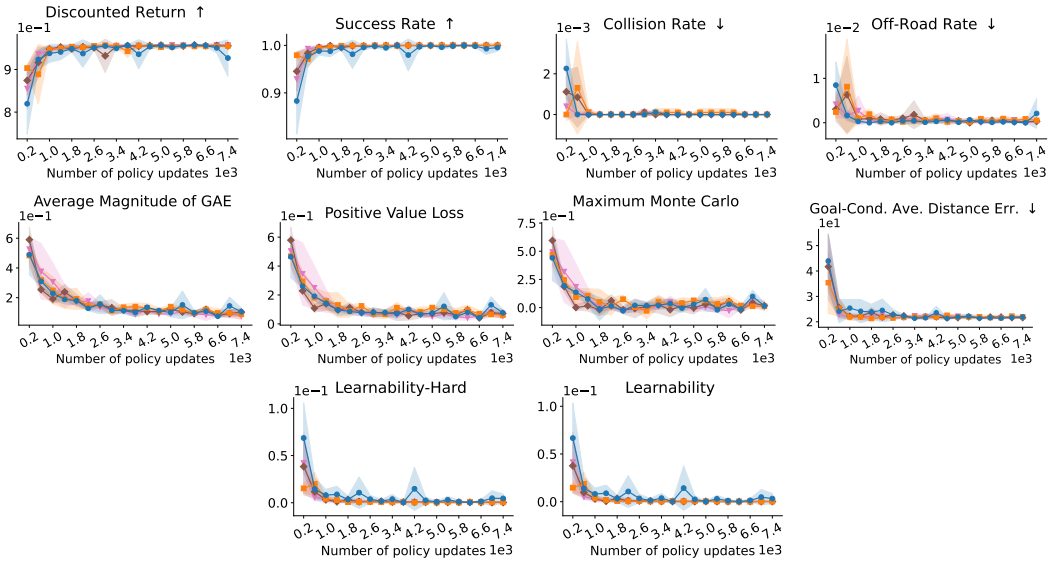


Figure 9: Case 3: Regret (U^{AMGAE}, U^{PVL}, U^{MaxMC}), realism ($U^{\text{GC-ADE}}$), and learnability (U^{Learn}, $U^{\text{Learn-hard}}$), progression during training with 80,000 scenarios from WOMD: We evaluate in 10,000 test scenarios. Bold markers indicate the mean, whereas the shaded area covers one standard deviation around it across two independent training runs.



(a) Evaluation on training partition



(b) Evaluation on test partition

Figure 10: Ablation: Performance, regret (U^{AMGAE}, U^{PVL}, U^{MaxMC}), realism ($U^{\text{GC-ADE}}$), and learnability (U^{Learn}, $U^{\text{Learn-hard}}$), progression during training for our ablation study on compute resources: We evaluate in **(a)** training partition, and **(b)** 150 test scenarios. Bold markers indicate the mean, whereas the shaded area covers one standard deviation around it across three training runs.

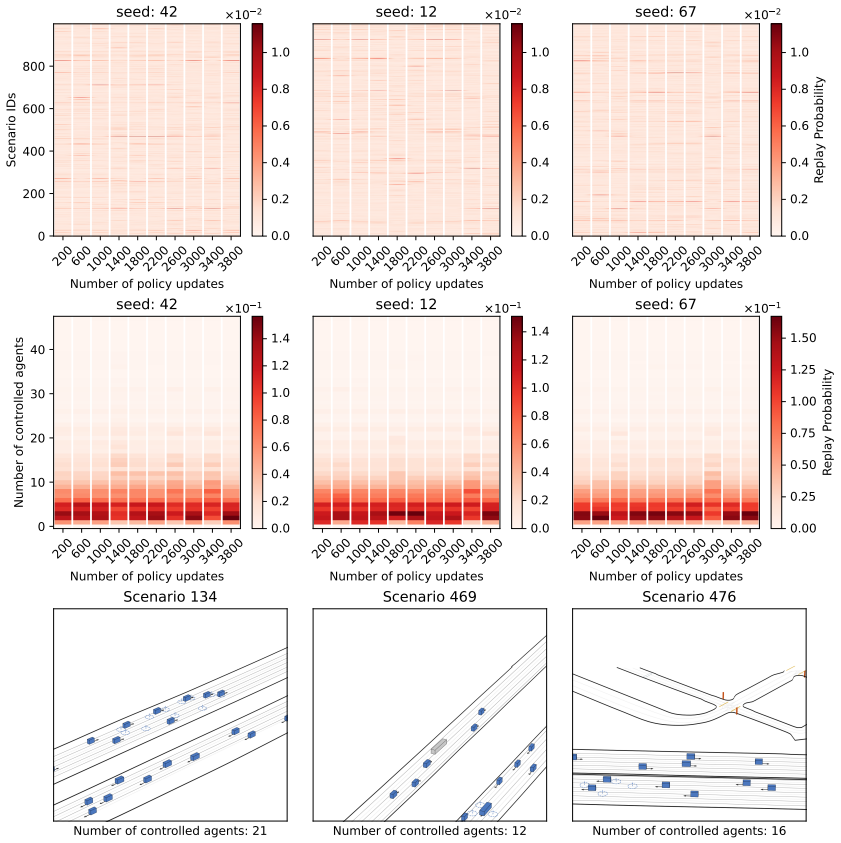


Figure 11: $\mathbb{P}_{\text{replay}}$ progression of PLR combined with $U^{\text{Act-MAE}}$ in mini WOMD: We illustrate **(top)** the evolution of $\mathbb{P}_{\text{replay}}$, where darker line segments indicate scenarios with higher replay likelihood, **(middle)** a version of replay distribution under categorization with respect to the number of controlled agents in scenarios, and **(bottom)** we exemplify three scenarios that appear frequently.

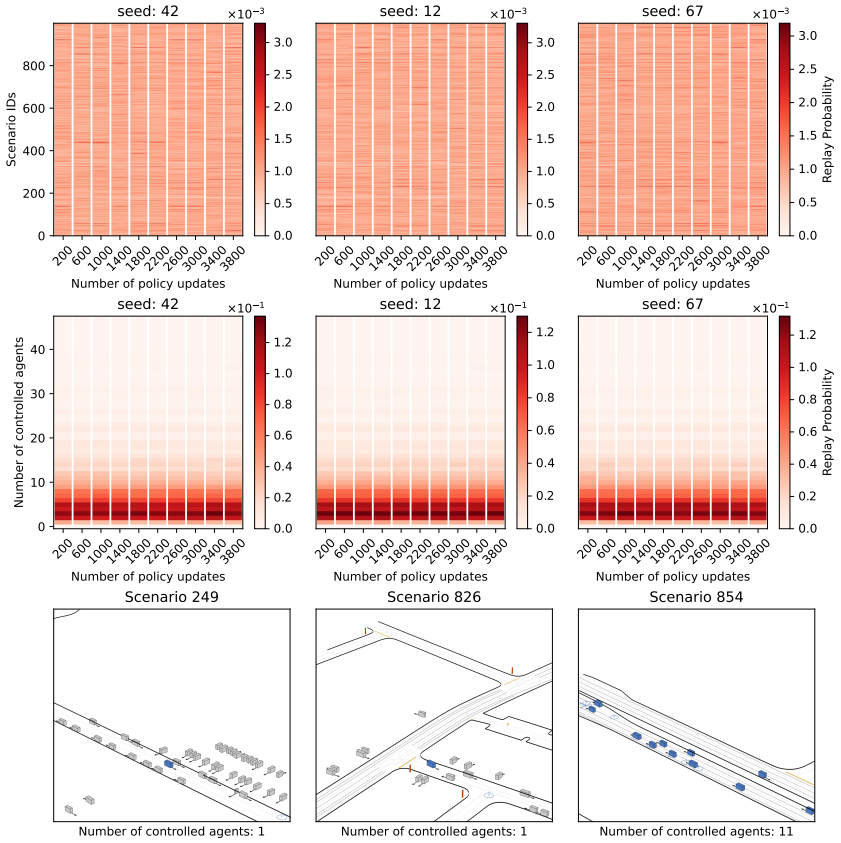


Figure 12: $\mathbb{P}_{\text{replay}}$ progression of PLR combined with U^{AMGAE} in mini WOMD: We illustrate **(top)** the evolution of $\mathbb{P}_{\text{replay}}$, where darker line segments indicate scenarios with higher replay likelihood, **(middle)** a version of replay distribution under categorization with respect to the number of controlled agents in scenarios, and **(bottom)** we exemplify three scenarios that appear frequently.

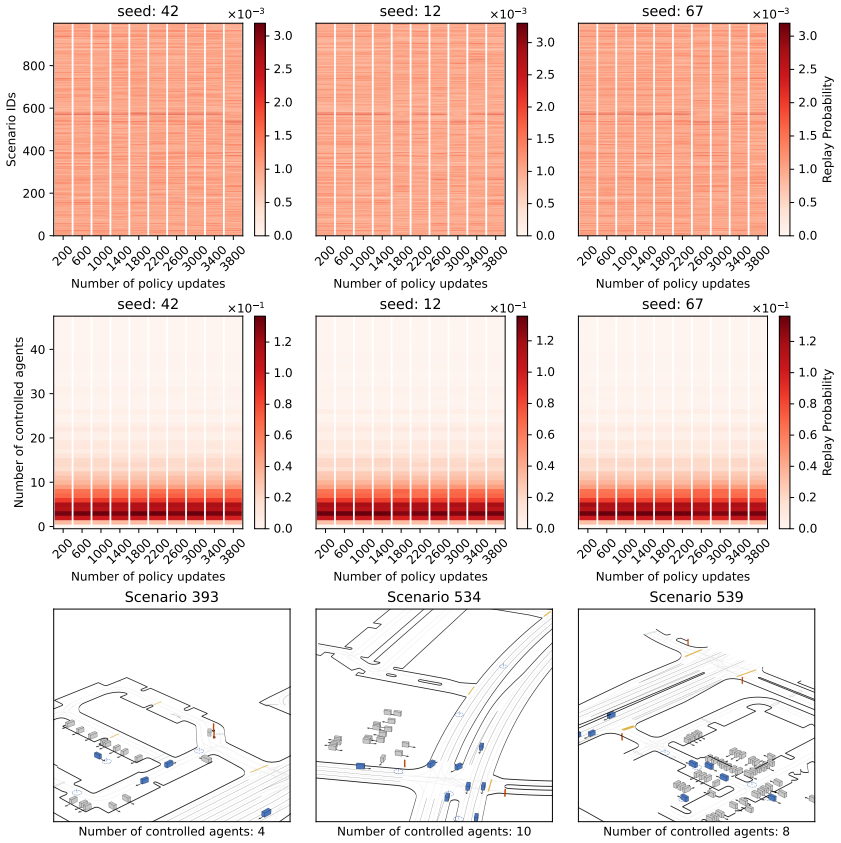


Figure 13: $\mathbb{P}_{\text{replay}}$ progression of PLR combined with $U^{\text{GC-ADE}}$ in mini WOMD: We illustrate **(top)** the evolution of $\mathbb{P}_{\text{replay}}$, where darker line segments indicate scenarios with higher replay likelihood, **(middle)** a version of replay distribution under categorization with respect to the number of controlled agents in scenarios, and **(bottom)** we exemplify three scenarios that appear frequently.

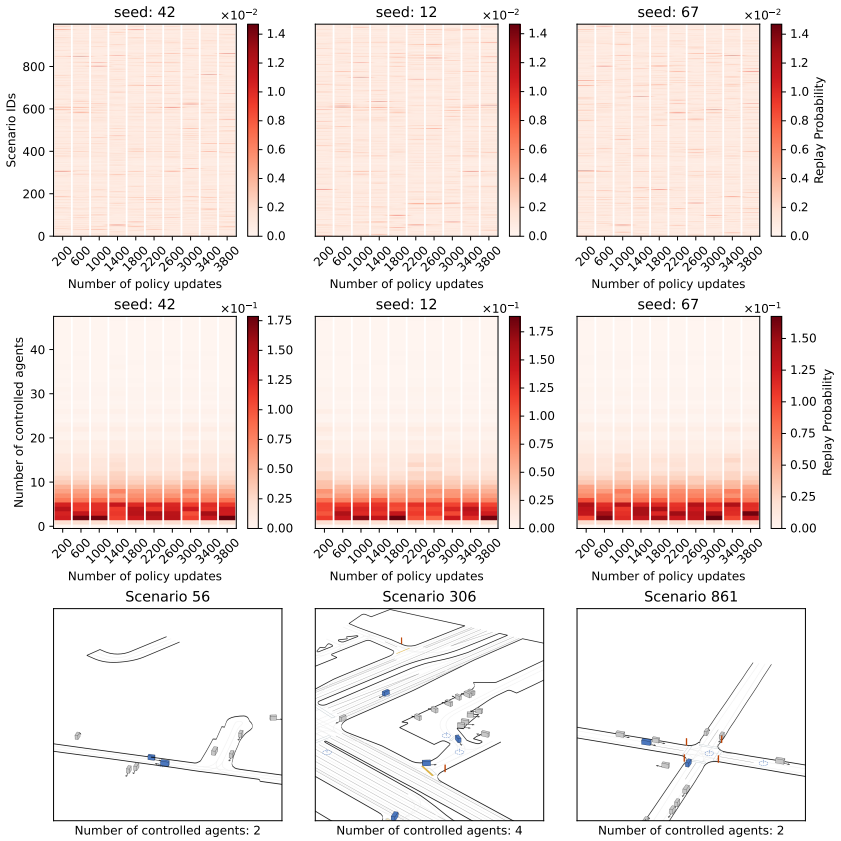


Figure 14: $\mathbb{P}_{\text{replay}}$ progression of PLR combined with $U^{\text{Learn-hard}}$ in mini WOMD: We illustrate **(top)** the evolution of $\mathbb{P}_{\text{replay}}$, where darker line segments indicate scenarios with higher replay likelihood, **(middle)** a version of replay distribution under categorization with respect to the number of controlled agents in scenarios, and **(bottom)** we exemplify three scenarios that appear frequently.

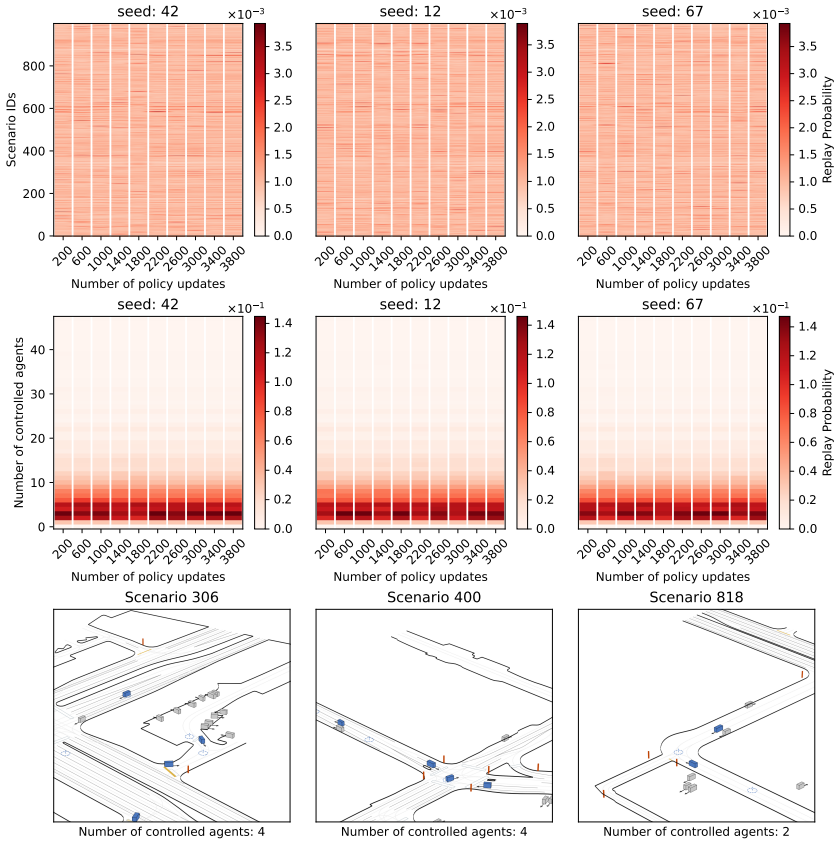


Figure 15: $\mathbb{P}_{\text{replay}}$ progression of PLR combined with U^{Learn} in mini WOMD: We illustrate **(top)** the evolution of $\mathbb{P}_{\text{replay}}$, where darker line segments indicate scenarios with higher replay likelihood, **(middle)** a version of replay distribution under categorization with respect to the number of controlled agents in scenarios, and **(bottom)** we exemplify three scenarios that appear frequently.

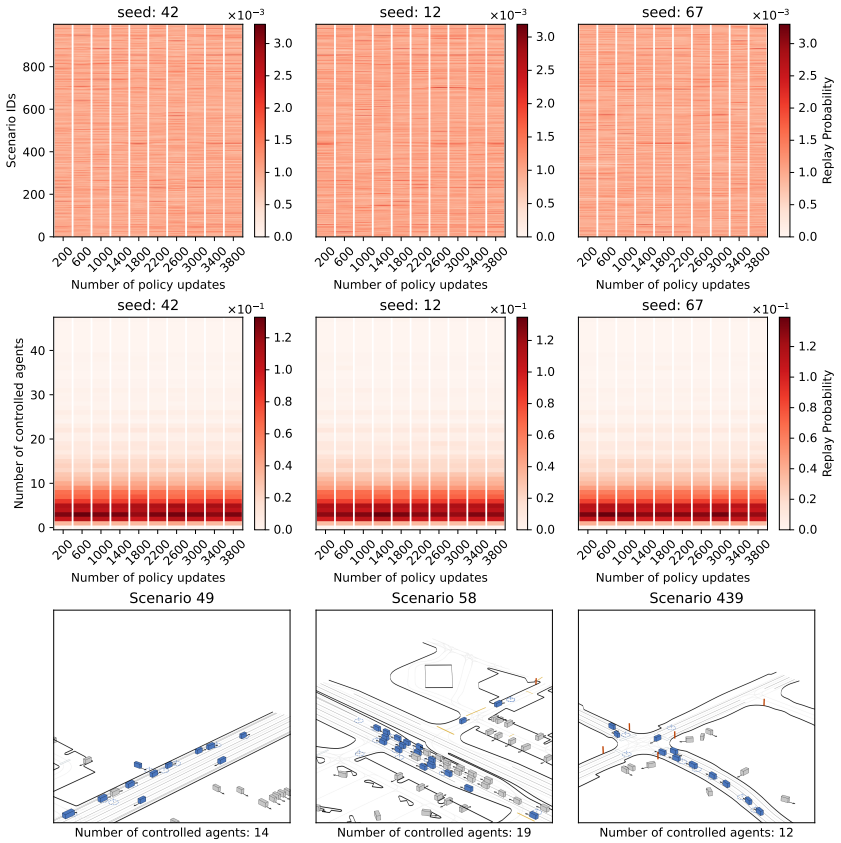


Figure 16: $\mathbb{P}_{\text{replay}}$ progression of PLR combined with U^{PVL} in mini WOMD: We illustrate **(top)** the evolution of $\mathbb{P}_{\text{replay}}$, where darker line segments indicate scenarios with higher replay likelihood, **(middle)** a version of replay distribution under categorization with respect to the number of controlled agents in scenarios, and **(bottom)** we exemplify three scenarios that appear frequently.

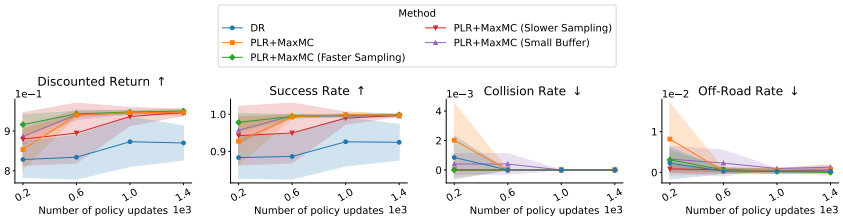


Figure 17: Sensitivity to sampling interval and buffer size: Performance progression of U^{MaxMC} in case 1 under three variations of our reported configuration, i.e., a buffer of 100 scenarios and scenario sampling intervals of 1×10^6 and 4×10^6 interactions. We evaluate on 150 test scenarios. Bold markers indicate the mean, whereas the shaded area covers one standard deviation around it across three independent training runs.

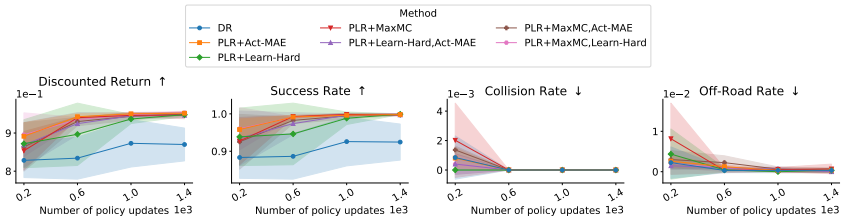


Figure 18: Combining utility functions across categories: Performance progression of three cross-category combinations in case 1, reported alongside their individual components and DR. We evaluate in 150 test scenarios. Bold markers indicate the mean, whereas the shaded area covers one standard deviation around it across three independent training runs.