Adversarial Environment Design via Regret-Guided Diffusion Models

Hojun Chung[1], **Junseo Lee**[1], **Minsoo Kim**[1], **Dohyeong Kim**[2], **and Songhwai Oh**[1,2,*]

[1]Interdisciplinary Program in Artificial Intelligence and ASRI, Seoul National University
[2]Department of Electrical and Computer Engineering and ASRI, Seoul National University
{hojun.chung, junseo.lee, minsoo.kim, dohyeong.kim}@rllab.snu.ac.kr,
songhwai@snu.ac.kr

Abstract

Training agents that are robust to environmental changes remains a significant challenge in deep reinforcement learning (RL). Unsupervised environment design (UED) has recently emerged to address this issue by generating a set of training environments tailored to the agent's capabilities. While prior works demonstrate that UED has the potential to learn a robust policy, their performance is constrained by the capabilities of the environment generation. To this end, we propose a novel UED algorithm, adversarial environment design via regret-guided diffusion models (ADD). The proposed method guides the diffusion-based environment generator with the regret of the agent to produce environments that the agent finds challenging but conducive to further improvement. By exploiting the representation power of diffusion models, ADD can directly generate adversarial environments while maintaining the diversity of training environments, enabling the agent to effectively learn a robust policy. Our experimental results demonstrate that the proposed method successfully generates an instructive curriculum of environments, outperforming UED baselines in zero-shot generalization across novel, out-of-distribution environments. Project page: https://rllab-snu.github.io/projects/ADD

1 Introduction

Deep reinforcement learning (RL) has achieved great success in various challenging domains, such as Atari [1], GO [2], and real-world robotics tasks [3, 4]. Despite the progress, the deep RL agent struggles with the generalization problem; it often fails in unseen environments even with a small difference from the training environment distribution [5, 6]. To train well-generalizing policies, various prior works have used domain randomization (DR) [7, 8, 9], which provides RL agents with randomly generated environments. While DR enhances the diversity of the training environments, it requires a large number of trials to generate meaningful structures in high-dimensional domains. Curriculum reinforcement learning [10, 11] has been demonstrated to address these issues by providing instructive sequences of environments. Since manually designing an effective curriculum for complicated tasks is challenging, prior works [12, 13] focus on generating curricula that consider the current agent's capabilities. Recently, unsupervised environment design (UED, [14]) has emerged as a scalable approach, notable for its advantage of requiring no prior knowledge. UED algorithms alternate between training the policy and designing training environments that maximize the regret of the agent. This closed-loop framework ensures the agent learns a minimax regret policy [15], assuming that the two-player game between the agent and the environment generator reaches the Nash equilibrium.

*Corresponding author: Songhwai Oh

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

There are two main approaches for UED: 1) learning-based methods, which employ an environment generator trained via reinforcement learning, and 2) replay-based methods, which selectively replay among previously generated environments. The learning-based methods [14, 16, 17] utilize an adaptive generator that controls the parameters that fully define the environment configuration. The generator receives a regret of the agent as a reward and is trained via reinforcement learning to produce environments that maximize the regret. While the learning-based methods can directly generate meaningful environments, training the generator with RL is unstable due to the moving manifold [16]. Additionally, we observe that the RL-based generator has limited environment coverage, which limits the generalization capability of the trained agent. In contrast, the replay-based methods [18, 19, 20] employ a random generator and select environments to revisit among previously generated environments. Since the random generator can produce diverse environments without additional training, they outperform the learning-based methods in zero-shot generalization tasks [20]. However, the replay-based methods are sample inefficient as they require additional episodes to evaluate the regret on the randomly generated environments.

In this work, we propose a sample-efficient and robust UED algorithm by leveraging the strong representation power of diffusion models [21]. First, to make UED suitable for using a diffusion model as a generator, we introduce soft UED, which augments the regret objective of UED with an entropy regularization term, as done in maximum entropy RL [22]. By incorporating the entropy term, we can ensure the diversity of the generated environments. Then, we present *adversarial environment design via regret-guided diffusion models* (ADD), which guides a diffusion-based environment generator with the regret of the agent to produce environments that are conducive to the performance improvement of the agent. Enabling this regret guidance requires the gradient of the regret with respect to the environment parameter. However, since the true value of the regret is intractable and the regret estimation methods used in prior works on UED are not differentiable, a new form of regret estimation method is needed. To this end, we propose a novel method that enables the estimation of the regret in a differentiable form by utilizing an environment critic, which predicts a return distribution of the current policy on the given environment. This enables us to effectively integrate diffusion models within the UED framework, significantly enhancing the environment generation capability.

Since the regret-guided diffusion does not require an additional training of the environment generator, we can preserve the ability to cover the high-dimensional environment domain as the random generator of the replay-based method. Moreover, ADD can directly generate meaningful environments via regret-guided sampling as the learning-based methods. By doing so, ADD effectively combines the strengths of previous UED methods while addressing some of their limitations. Additionally, unlike other UED methods, ADD allows us to control the difficulty levels of the environments it generates by guiding the generator with the probability of achieving a specific return. It enables the reuse of the learned generator in various applications, such as generating benchmarks.

We conduct extensive experiments across challenging tasks commonly used in UED research: partially observable maze navigation and 2D bipedal locomotion over challenging terrain. Experimental results show that ADD achieves higher zero-shot generalization performance in unseen environments compared to the baselines. Furthermore, our analysis on the generated environments demonstrates that ADD produces an instructive curriculum with varying complexity while covering a large environment configuration space. As a result, it is shown that the proposed method successfully generates adversarial environments and facilitates the agent to learn a policy with solid generalization capabilities.

2 Related Work

2.1 Unsupervsied Curriculum Reinforcement Learning

While curriculum reinforcement learning [13, 23, 24] has been shown to enhance the generalization performance of the RL agent, Dennis et al. [14] first introduce the concept of the unsupervised environment design (UED). UED encompasses various environment generation mehods, such as POET [12, 25] and GPN[26]. In this work, we follow the original concept of UED, which aims to learn a minimax regret policy [15] by generating training environments that maximize the regret of the agent. Based on this concept, the learning-based methods train an environment generator via reinforcement learning. PAIRED [14] estimates the regret with a difference between returns

obtained by two distinct agents, and trains RL-based generator by utilizing the regret as a reward. Recently, CLUTR [16] and SHED [17] utilize generative models to improve the performance of PAIRED. CLUTR trains the environment generator on the learned latent space, and SHED supplies the environment generator with augmented experiences created by diffusion models. Despite the progress, training the generator via RL is unstable due to the moving manifold [16, 27] and often struggles to generate diverse environments. On the other hand, replay-based methods based on PLR [18] utilize a random environment generator and decide which environments to replay. ACCEL [20] combines the evolutionary approaches [12, 25] and PLR by taking random mutation on replayed environments. While these replay-based methods show scalable performance on a large-scale domain [28] and outperform the learning-based methods, they do not have the ability to directly generate meaningful environments. Unlike prior UED methods, we augment the regret objective of UED with an entropy regularization term and propose a method that employs a diffusion model as an environment generator to enhance the environment generation capability. Our work is also closely related to data augmentation for training robust policy. Particularly, DRAGEN [29] and ISAGrasp [30] augment existing data in learned latent spaces to train a policy that is robust to unseen scenarios. Our algorithm, on the other hand, focuses on generating curricula of environments without any prior knowledge and dataset.

2.2 Diffusion Models

Diffusion models [21, 31, 32] have achieved remarkable performance in various domains, such as image generation [33], video generation [34], and robotics [35, 36, 37]. Particularly, diffusion models effectively perform conditional generation using guidance to generate samples conditioned on class labels [38, 39] or text inputs [40, 41, 42]. Prior works also guide the diffusion models utilizing an additional network or loss functions, such as adversarial guidance to generate images to attack a classifier [43], safety guidance using pre-defined functions to generate safety-critical driving scenarios [44], and guidance using reward functions trained by human preferences to generate censored samples. [45]. We note that our implementation of the regret-guided diffusion model is based on the work of Dhariwal et al. [38] and Yoon et al. [45].

3 Background

3.1 Unsupervised Environment Design

Unsupervised environment design (UED, [14]) aims to provide an adaptive curriculum to learn a policy that successfully generalizes to various environments. The environments are represented using a Markov decision process (MDP), defined as $\langle A, S, \mathcal{P}, \mathcal{R}, \rho_0, \gamma \rangle$, where A is a set of actions, S is a set of states, $\mathcal{P} : S \times A \times S \to [0, 1]$ is a transition model, $\mathcal{R} : S \times A \to \mathbb{R}$ is a reward function, $\rho_0 : S \to [0, 1]$ is an initial state distribution and γ is a discount factor. UED employs an environment generator that designs environments by controlling free environment parameters of underspecified environments, which is represented using an underspecified Markov decision process (UMDP). UMDP is defined as $\mathcal{M} = \langle A, S, \Theta, \mathcal{P}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, \rho_0^{\mathcal{M}}, \gamma \rangle$, where Θ is a set of free environment parameters. Assigning a value to the free environment parameter $\theta \in \Theta$ results in a specific MDP $\mathcal{M}_\theta$ with the environment configuration ($\mathcal{P}^\theta = \mathcal{P}^{\mathcal{M}}(\theta), \mathcal{R}^\theta = \mathcal{R}^{\mathcal{M}}(\theta), \rho_0^\theta = \rho_0^{\mathcal{M}}(\theta)$). For example, when learning a mobile robot to navigate towards the goal point while avoiding obstacles, θ could represent the positions of obstacles, the position of the goal, and the start position of the robot.

UED algorithms alternate between designing a set of environments and training the agent on the generated environments. The environment generator first produces an environment parameter θ that maximizes the agent's regret. The regret of the policy π on environment $\mathcal{M}^\theta$ is defined as,

$$\textsc{Regret}(\pi, \theta) := -V(\pi, \theta) + \max_{\pi' \in \Pi} V(\pi', \theta), \tag{1}$$

where Π is a set of policies and $V(\pi, \theta) := \mathbb{E}_{\rho_0^\theta, \pi, \mathcal{P}^\theta}\left[\sum_{n=0}^{N} r_n \gamma^n\right]$ is an expected return where r_n is a reward obtained by π at timestep n on $\mathcal{M}^\theta$. Then, the agent is trained on the generated environment to maximize the expected return, resulting in minimizing the regret. This framework can be formulated with the following two-player minimax game:

$$\min_{\pi \in \Pi} \max_{\theta \in \Theta} \textsc{Regret}(\pi, \theta). \tag{2}$$

It is ensured that the agent learns the minimax regret policy $\pi^* \in \underset{\pi \in \Pi}{\operatorname{argmin}} \max_{\theta \in \Theta} \textsc{Regret}(\pi, \theta)$ by assuming the two-player game (2) reaches the Nash equilibrium [14, 19]. However, learning the minimax regret policy is challenging. Since the objective (2) does not guarantee the diversity of generated environments, the agent may not be trained on sufficiently various environments.

3.2 Diffusion Probabilistic Models

A diffusion probabilistic model [21] is a generative model that generates samples from noise via iterative denoising steps. Diffusion models start with perturbing data by progressively adding Gaussian noise, called the *forward process*. The forward process can be modeled with a value-preserving stochastic differential equation (VP SDE, [31]):

$$dX_t = -\frac{\beta_t}{2} X_t dt + \sqrt{\beta_t} dW_t, \tag{3}$$

where $t \in [0, T]$ is a continuous diffusion time variable, $\beta_t > 0$ is a variance schedule, and W_t is a standard Brownian motion. Since the forward process (3) has tractable conditional marginal distributions $p_t(X_t|X_0) = \mathcal{N}(\sqrt{\alpha_t} X_0, (1-\alpha_t) I)$ where $\alpha_t = e^{-\int_0^t \beta_t \, dt}$, $p_T(X_T)$ will be corrupted into $\mathcal{N}(0, I)$ when $T \to \infty$.

Generating samples following the data distribution $p_{data}(\cdot)$ requires a *reverse process*, a reverse-time SDE that has the same marginal distributions as the forward process (3). By Anderson's theorem [46], the reverse process can be formulated with a reverse-time SDE defined as,

$$dX_t = -\beta_t \left[\frac{1}{2} X_t + \nabla_{X_t} \log p_t(X_t) \right] dt + \sqrt{\beta_t} dW_t. \tag{4}$$

Hence, learning a diffusion model means learning a score network $s_\phi(X_t, t)$ that approximates a score function $\nabla_{X_t} \log p_t(X_t)$. The score network is trained via score matching [47], then plugged into the reverse process (4):

$$dX_t = -\beta_t \left[\frac{1}{2} X_t + s_\phi(X_t, t) \right] dt + \sqrt{\beta_t} dW_t. \tag{5}$$

Indeed, we can generate samples by solving the approximated reverse process (5) with an initial condition $X_T \sim \mathcal{N}(0, I)$.

To generate samples with label Y using the diffusion model, the score function of the conditional distribution $p_t(X_t|Y)$ should be estimated. Since $p_t(X_t|Y) \propto p_t(X_t, Y) = p_t(X_t) p_t(Y|X_t)$ due to Bayes' rule, conditional samples can be generated by solving the reverse process with classifier guidance [38]:

$$\begin{aligned} \hat{s}_\phi(X_t, t) &= s_\phi(X_t, t) + \omega \nabla_{X_t} \log \hat{p}_t(Y|X_t), \\ dX_t &= -\beta_t \left[\frac{1}{2} X_t + \hat{s}_\phi(X_t, t) \right] dt + \sqrt{\beta_t} dW_t, \end{aligned} \tag{6}$$

where $\hat{p}_t(Y|X_t)$ is a time-dependent classifier network and $\omega > 0$ is a guidance weight to scale classifier gradients.

4 Proposed Method

In this section, we describe our approach to employ a diffusion model as an environment generator to enhance the environment generation capability. We first introduce soft UED, which mutates UED to be more suitable for using a diffusion model as a generator by augmenting the original objective with the entropy regularization term. Then, we propose a novel soft UED algorithm, adversarial environment design via regret-guided diffusion models (ADD). ADD consists of two key components: 1) a diffusion-based environment generation by using the regret as a guidance, and 2) a method to estimate the regret in a differentiable form. We present these key components in detail and conclude the section with an explanation of the overall system and its advantages compared to prior UED methods.

4.1 Soft Unsupervised Environment Design

In this section, we introduce soft UED, designed to guarantee the diversity of environments by adding an entropy regularization term to the original UED objective (2). Soft UED is defined as the following minimax game between the agent and the environment generator:

$$\min_{\pi\in\Pi}\max_{\Lambda\in\mathcal{D}_\Lambda}\mathop{\mathbb{E}}_{\theta\sim\Lambda}\left[\textsc{Regret}(\pi,\theta)\right]+\frac{1}{\omega}H(\Lambda), \tag{7}$$

where Λ is a distribution over Θ, $\mathcal{D}_\Lambda$ is a set of distributions over Θ, $H(\Lambda) := -\sum_{\theta\in\Theta}\Lambda(\theta)\log\Lambda(\theta)$ is an entropy of Λ, and ω is a regularization coefficient. Based on the fact that H is concave, we can show that the strong duality holds:

Proposition 4.1. *Let* $L(\pi,\Lambda) := \mathbb{E}_{\theta\sim\Lambda}\left[\textsc{Regret}(\pi,\theta)\right]+\frac{1}{\omega}H(\Lambda)$ *and assume that* S, A, *and* Θ *are finite. Then,* $\min_{\pi\in\Pi}\max_{\Lambda\in\mathcal{D}_\Lambda}L(\pi,\Lambda)=\max_{\Lambda\in\mathcal{D}_\Lambda}\min_{\pi\in\Pi}L(\pi,\Lambda)$.

The proof is detailed in Appendix A.1. Proposition 4.1 implies that there exists a valid optimal point $(\tilde{\pi},\tilde{\Lambda})$, and it is stationary for alternative optimization of π and Λ. Hence, the agent will learn a soft minimax regret policy $\tilde{\pi}=\operatorname{argmin}_{\pi\in\Pi}\max_{\Lambda\in\mathcal{D}_\Lambda}L(\pi,\Lambda)$ if it reaches the optimal point. One of the most significant difference from the original UED is the role of the environment generator. Instead of finding a specific environment parameter that maximizes the regret, soft UED updates the environment generator to sample environment parameters from a distribution that maximizes the objective function of soft UED.

We note that the soft UED framework encompasses prior UED algorithms. In the learning-based methods, the generator is trained with RL using an entropy bonus, which is known to enhance performance [48] and plays a similar role to $H(\Lambda)$. The replay-based methods also consider the diversity of environments by sampling environment parameters from a probability distribution proportional to the regret, instead of selecting a parameter that maximizes the regret. Therefore, soft UED can be considered as a general framework that incorporates practical methods.

4.2 Regret-Guided Diffusion Models

Soft UED converts the problem of generating regret-maximizing environments into a problem of sampling the environment parameter θ from a desired distribution $\Lambda^\pi := \operatorname{argmax}_{\Lambda\in\mathcal{D}_\Lambda}L(\pi,\Lambda)$. It is a well-known fact that Λ^π has a closed-form solution as follows:

$$\Lambda^\pi(\theta)=\frac{u(\theta)\exp(\omega\textsc{Regret}(\pi,\theta))}{C^\pi}, \tag{8}$$

where C^π is a normalizing constant, and $u(\cdot)$ denotes an uniform distribution over Θ. Inspired by the classifier guidance (6), we solve this sampling problem by guiding a pre-trained diffusion model with the regret. To this end, we decompose the score function of Λ^π as follows:

$$\nabla_{\theta_t}\log\Lambda^\pi_t(\theta_t)=\nabla_{\theta_t}\log u_t(\theta_t)+\omega\nabla_{\theta_t}\textsc{Regret}_t(\pi,\theta_t), \tag{9}$$

where t is a diffusion time variable, θ_t is an environment parameter perturbed by the forward process (3), $u_t(\cdot)$ denotes a distribution of θ_t when $\theta_0\sim u(\cdot)$, $\Lambda^\pi_t(\cdot)$ denotes a distribution of θ_t when $\theta_0\sim\Lambda^\pi(\cdot)$, and $\textsc{Regret}_t(\pi,\theta_t)$ is a time-dependent regret on the noised environment θ_t, which is equal to $\textsc{Regret}(\pi,\theta_0)$. We approximate the first term $\nabla_{\theta_t}\log u_t(\theta_t)$ with a score network $s_\phi(\theta_t,t)\approx\nabla_{\theta_t}\log u_t(\theta_t)$ that is learned by training a diffusion-based environment generator on the randomly generated environment dataset before the agent begins to learn. Then, we can formulate a regret-guided reverse process with a reverse-time SDE as follows:

$$\begin{aligned}s^\pi_\phi(\theta_t,t)&=s_\phi(\theta_t,t)+\omega\nabla_{\theta_t}\textsc{Regret}_t(\pi,\theta_t),\\ d\theta_t&=-\beta_t\left[\frac{1}{2}\theta_t+s^\pi_\phi(\theta_t,t)\right]dt+\sqrt{\beta_t}dW_t.\end{aligned} \tag{10}$$

Hence, if a gradient of the regret is tractable, we can sample an environment parameter θ_0 from the desired distribution Λ^π by solving the regret-guided reverse process (10) with an initial condition $\theta_T\sim\mathcal{N}(0,I)$. However, the regret (1) is intractable since we cannot access the environment-specific optimal policy. Prior works on UED propose various methods to estimate the regret using episodic returns or temporal difference errors, but none of them are differentiable w.r.t. θ_t since agents cannot access the environment parameter and the reward function.

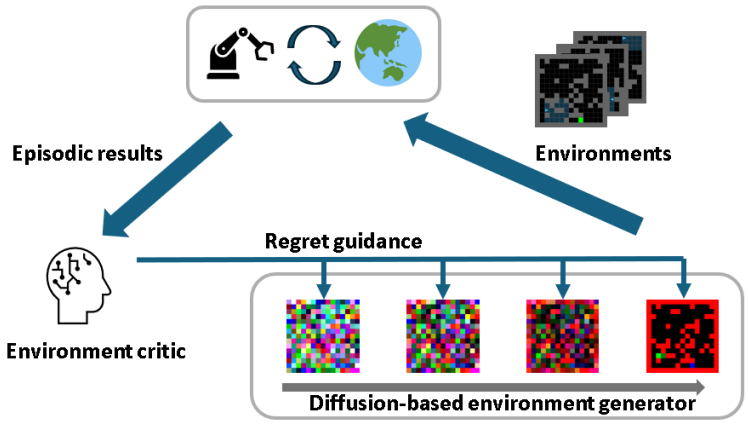

Figure 1: **Overview of ADD.** After the agent is trained on environments produced by the environment generator, the environment critic is updated using the episodic results. Then, the environment critic guides the diffusion-based environment generator with the regret to produce adversarial environments. By repeating this process, the agent learns a policy that is robust to environmental changes.

4.3 A Differentiable Regret Estimation

In order to estimate the regret in a differentiable form, we present a novel method based on a flexible regret [14], which is known to enhance the performance of the learning-based UEDs [16]. The main idea is to estimate the regret with a difference between the maximum and average episodic returns that can be achieved by the agent. To make it differentiable w.r.t. θ_t, we utilize an environment critic that predicts the return of the agent in the given environment parameter, as done in DSAGE [49] and LPG [50]. The environment critic τ_ψ learns to predict a distribution of returns, analogous to distributional RL [51], to better capture the stochasticity of the environment and policy. Based on a support defined as $\{z_i = v_{min} + \frac{i}{M-1}(v_{max} - v_{min})\}_{i=0}^{M-1}$, which is a set of centers of M bins that evenly divide the return domain $[v_{min}, v_{max}]$, we obtain an estimated categorical return distribution from an environment critic output $l(\theta_t, t; \psi) \in \mathbb{R}^M$ as follows:

$$\hat{\mathcal{Z}}_\pi(\theta_t, t) = z_i \quad \text{w.p.} \quad \frac{\exp(l_i(\theta_t, t; \psi))}{\sum_{j=0}^{M-1} \exp(l_j(\theta_t, t; \psi))}. \tag{11}$$

To align $\hat{\mathcal{Z}}_\pi$ with a true return distribution, we train the environment critic by gradient descent on the cross entropy loss between $\hat{\mathcal{Z}}_\pi(\theta_t, t)$ and a target distribution, which is constructed by projecting episodic returns that the agent achieves on the environment $\mathcal{M}^{\theta_0}$ onto the support $\{z_i\}_{i=0}^{M-1}$.

After the environment critic is updated, we estimate the regret (1) with a difference between the maximum return that the current agent can achieve and average of the predicted return distribution. However, the process of finding a maximum achievable return from the distribution is not differentiable. To address this issue, we further approximate the maximum with a conditional value at risk (CVaR), based on the fact that $\text{CVaR}_\alpha(\hat{\mathcal{Z}})$ converges to the maximum as a risk measure α goes zero. As a result, we estimate the regret of the agent as follows:

$$\textsc{Regret}_t(\theta_t, t) \approx \text{CVaR}_\alpha(\hat{\mathcal{Z}}_\pi(\theta_t, t)) - \mathbb{E}(\hat{\mathcal{Z}}_\pi(\theta_t, t)). \tag{12}$$

4.4 Adversarial Environment Design via Regret-Guided Diffusion Models

An overview of ADD is provided in Figure 1. First, a diffusion-based environment generator, which is pre-trained on the randomly generated environment dataset, produces a set of environments for the agent. After the agent interacts with the generated environments and is trained via reinforcement learning, the episodic results are utilized to update the environment critic. Then, the environment critic estimates the regret of the agent in a differentiable form (12) and guides the reverse process of the diffusion-based environment generator (10), resulting in environment parameters following the

distribution that maximizes the soft UED objective (7). By repeating this process, the agent learns the soft minimax regret policy, which is robust to the variations of environments. A pseudocode of the algorithm is shown in Appendix A.4.

Since ADD does not require an additional training of the pre-trained diffusion model, the ability to cover the high-dimensional environment domain can be preserved. Furthermore, ADD enables the generator to directly produce meaningful environments via regret-guided reverse process. Therefore, ADD can be seen as having both the advantage of the replay-based methods and learning-based methods while resolving some of their limitations. Additionally, we can control the difficulty level of the generated environments after the training of the RL agent is over. In detail, we can generate environments of difficulty level $k \in \{1, 2, \dots, M\}$ by replacing the regret in the regret-guided reverse process (10) with a log probability of achieving a specific return z_{M-k} as follows:

$$\begin{aligned} s'_\phi(\theta_t, t) &= s_\phi(\theta_t, t) + \omega \nabla_{\theta_t} \log \Pr(\hat{\mathcal{Z}}_\pi(\theta_t, t) = z_{M-k}), \\ d\theta_t &= -\beta_t \left[\frac{1}{2}\theta_t + s'_\phi(\theta_t, t) \right] dt + \sqrt{\beta_t} dW_t. \end{aligned} \tag{13}$$

It enables the reuse of the learned generator and environment critic in various applications, such as generating benchmarks with varying difficulties.

5 Experiments

Tasks We conduct extensive experiments with two challenging tasks. First, we evaluate the proposed method on a partially observable navigation task with a discrete action space and sparse rewards. Then, we assess the performance of our algorithm on a 2D bipedal locomotion task, which has a continuous action space while offering dense rewards.

Baselines We compare the proposed method against several UED baselines. For the learning-based method, we use PAIRED [14], which trains the environment generator via reinforcement learning. For the replay-based method, we use PLR$^\perp$ [19], which utilizes the random generator and updates the agent only with episodes from the replayed environments. To benchmark performance, we use ACCEL [20], a current state-of-the-art UED algorithm that applies random mutations to replayed environments. Among the two implementation methods of the ACCEL, we use the one that samples environment parameters from the full parameter range, rather than the one that restricts the sampling to a range that ensures simple environments are generated, as the latter could be seen as incorporating prior knowledge. Domain randomization (DR) is also included in baselines so that we can demonstrate the effectiveness of UED. Lastly, we use ADD w/o guidance to show whether the regret guidance induces the diffusion model to generate adversarial environments and enhances the performance of the agent.

Outline We first train a diffusion-based environment generator on the randomly generated environment dataset. Then, we use proximal policy optimization (PPO, [52]) to train the agent on the environments generated by UED methods. To evaluate the generalization capability of the trained agent, we measure the zero-shot transfer performance in challenging, human-designed environments. Additionally, to understand where the differences in performance originate, we conduct quantitative and qualitative analyses on the curriculum of the generated environments by tracking complexity metrics and drawing t-SNE plots. For space consideration, we elaborate on detailed experimental settings including environment parameterization methods in Appendix B.

5.1 Partially Observable Navigation Task

We first evaluate the proposed method on a maze navigation task [14], which is based on the Minigrid [53]. In this task, an agent is trained to take a discrete action using an observation from its surroundings to receive a sparse reward upon reaching a goal. For prior UED methods, we set the maximum number of blocks in a grid environment to 60, aligning with Parker-Holder et al. [20]. For the proposed method, we train the diffusion-based environment generator on 10M random environments whose number of blocks uniformly varies from zero to 60. Then, we train the LSTM-based policy for 250M environmental steps and evaluate the zero-shot performance on 12 challenging test environments from prior works [14, 19].

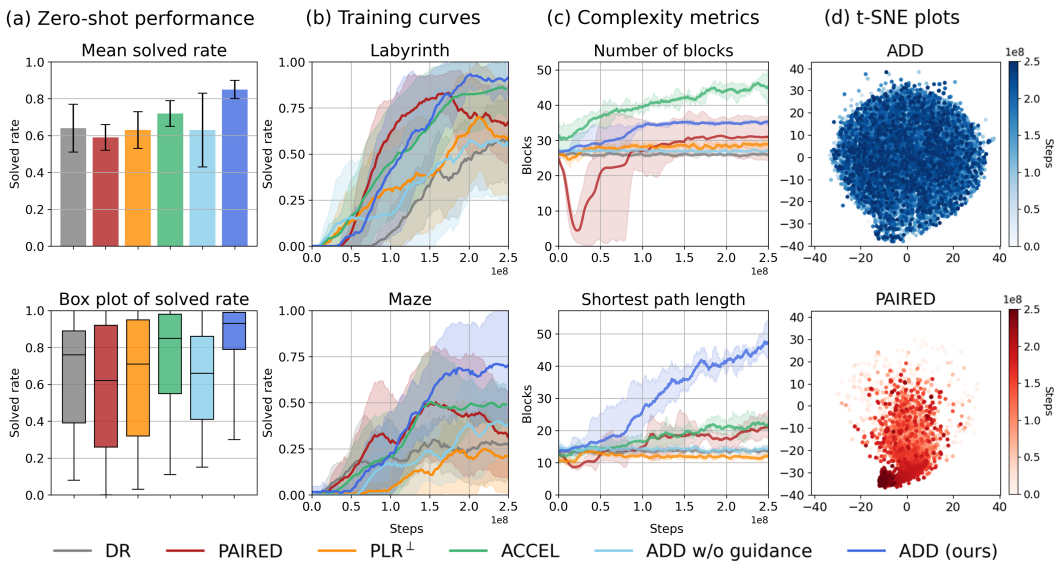

Figure 2: **Partially observable navigation results.** **(a)**: Zero-shot performance on the 12 test environments. We report results across five random seeds, each evaluated over 100 independent episodes per environment. **(b)**: Training curves on two challenging test environments. **(c)**: Complexity metrics of the generated environments during training. **(d)**: t-SNE embedding of the generated environments during training.

Performance. In Figure 2(a), we report the mean solved rate and box plot to compare the zero-shot performance of the learned policy on the test environments. The result demonstrates that ADD outperforms the baseline methods while achieving 85% of the mean solved rate, which is 18% higher compared to the ACCEL. Furthermore, ADD achieves the highest Q1, Q2, and Q3 values while its interquartile range, defined as Q3 - Q1, is 51% smaller than ACCEL. Therefore, we can infer that the proposed method consistently outperforms the baselines in the challenging test environments. In Figure 2(b), we report training curves on two test environments consisting of Maze and Labyrinth. The results demonstrate that ADD shows the monotonic performance improvement and achieves a higher solved rate compared to other baselines. While ACCEL shows 13% higher mean solved rate than DR, PAIRED and PLR$^{\perp}$ do not show notable performance improvement by applying UED techniques. Particularly, PAIRED shows 8% lower mean solved rate compared to DR, demonstrating the challenge of the learning-based UED methods. ADD w/o guidance shows 63% of mean solved rate similar to DR, demonstrating that the regret guidance is critical for training the robust policy. Please refer to Appendix C.1 for detailed per-environment results.

We note that the performance is measured after the fixed number of environmental steps, in line with some UED papers [19, 54] and traditional deep reinforcement learning research. In contrast, Parker-Holder et al. [20] recorded performance after the fixed number of policy updates. Since the replay-based methods require additional episodes without policy updates, using the number of environmental steps may be seen as unfair to PLR$^{\perp}$ and ACCEL. To address this issue, we also measured the performance of our method trained with only half the environmental steps, aligning the number of policy updates with PLR$^{\perp}$ and ACCEL. When using half the environmental steps, ADD achieves a 72% mean solved rate, which ties with ACCEL. This demonstrates that the proposed method remains competitive, even when using the number of policy updates as a metric.

Generated curriculum. In Figure 2(c), we report complexity metrics consisting of the number of blocks and shortest path length. The results demonstrate that complexity metrics of ADD w/o guidance and DR are almost consistent over time since they do not consider the policy. While PLR$^{\perp}$ eventually generates environments with a larger number of blocks compared to DR, ADD and PAIRED generate a curriculum with significantly increasing complexity. Specifically, ADD eventually generates environments with the second largest number of blocks and the longest shortest path. ACCEL also shows significantly increasing complexities despite being based on PLR$^{\perp}$. This is because only up to 60 blocks exist on the 13X13 grid, so random mutation increases the expectation

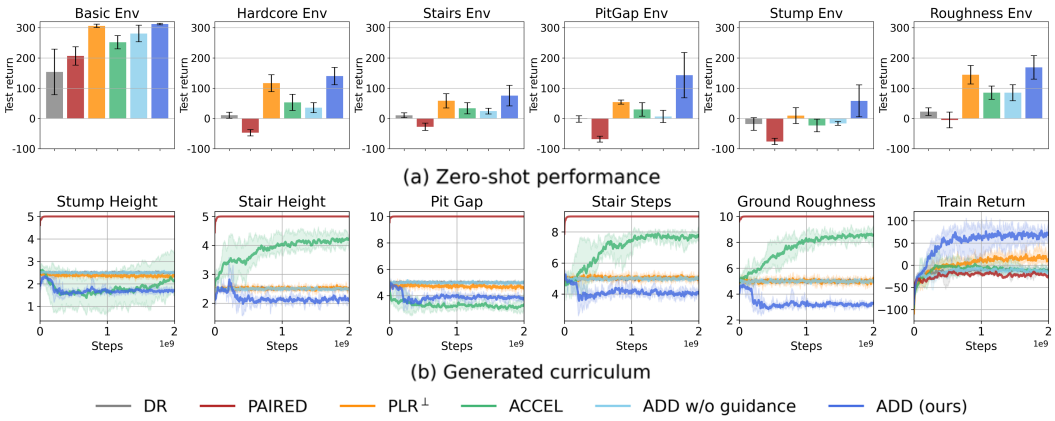

Figure 3: **2D bipedal locomotion task results. (a)**: Zero-shot performance on the six test environments. We report results across five random seeds, each evaluated over 100 independent episodes per environment. **(b)**: Complexity metrics of the generated environments and episodic return achieved during training.

of the number of blocks. Therefore, it can be seen that ADD and PAIRED efficiently generate complex environments by adapting to the current policy while PLR$^{\perp}$ struggles to find environments with high complexity. To compare the distribution of generated environments, we report t-SNE [55] plots in Figure 2(d). While the environments generated by PAIRED eventually cover only a specific region, the environments generated by ADD cover a significantly larger region over time. The results demonstrate that ADD successfully generates adversarial environments while preserving diversity.

5.2 2D Bipedal Locomotion Task

We evaluate the proposed method on the 2D bipedal locomotion task, which is based on the BipedalWalker task in OpenAI Gym [56] and adopted by Parker Holder et al. [20]. In this task, an agent is trained to control its four motors using observation from its Lidar sensor and joints to walk over challenging terrain. UED methods, including the proposed algorithm, need to provide environment parameters consisting of stump height, stair height, pit gap, stair steps, and ground roughness. We note that each parameter increases the difficulty of the environment as its value increases. We train the RL agent for two billion environmental steps and evaluate the zero-shot performance on six test environments.

Performance. Figure 3(a) shows the average return on each test environment. The proposed algorithm achieves the highest return across all environments, with an average of 149.6. Even with half the environmental steps, it achieves a score of 127.4, still surpassing PLR$^{\perp}$. ACCEL shows lower performance than PLR$^{\perp}$, which can be attributed to the lower sample efficiency induced by the additional interaction between the environment and the agent to assess the modified environments. On the other hand, PAIRED achieves the lowest return in all test environments except the easiest Basic Environment. This shows that the learning-based methods struggle to train a robust policy in practice. We note that a recent work [48] stabilizes the training of PAIRED in this task by integrating the evolutionary concept of ACCEL. While applying the evolutionary approach to ADD is possible, we leave it for future work. Lastly, ADD w/o guidance demonstrates superior generalization performance compared to DR. Although these two methods are theoretically identical, this difference is presumably caused by the limited size of the dataset used for training the diffusion-based environment generator.

Generated curriculum Figure 3(b) presents the complexity metrics of the generated training environments and the episodic returns achieved by the RL agent. Unlike the partially observable navigation task, the complexity measure of the environments generated by ADD gradually decreases. This result arises since the randomly generated environments are excessively challenging for the current agent. As evidence, examining the returns achieved by the agent in the generated environments reveals that all methods, except for ADD, consistently yield returns of 0 or below. From these results, we can infer that the proposed algorithm generates environments that are not merely more difficult but are

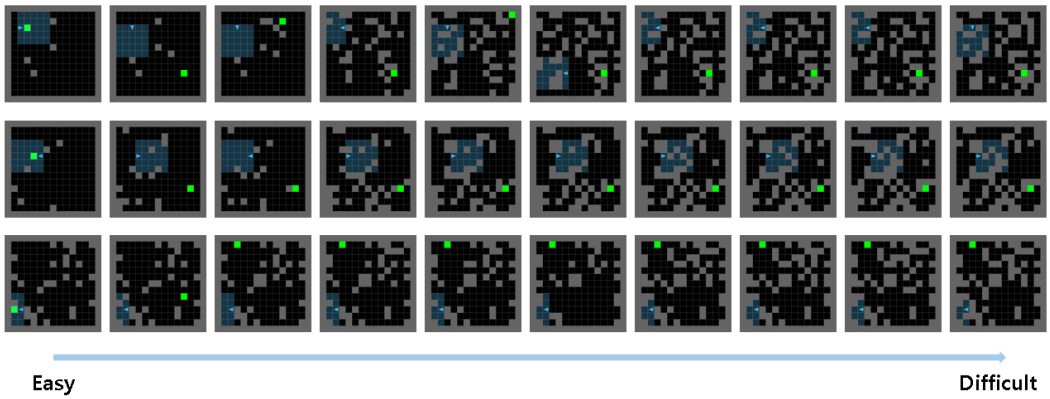

Figure 4: **Controllable generation results for the partially observable navigation task**. The figure shows the results of guiding the generator to generate progressively more difficult environments. We note that each row is generated from the same initial noise θ_T.

conducive to the agent's learning process. For detailed experimental results including a qualitative analysis on the generated environments, please refer to Appendix C.2.

5.3 Controllable Generation

To demonstrate the ability to control the difficulty level of the generated environments, we provide the example environment generation results in Figure 4. We control the difficulty level k by guiding the diffusion-based environment generator with a log probability of achieving a specific return z_{M-k}, as described in (13). We vary k from zero to $M-1$ so that the difficulty level of generated environments increases. Environments generated with $k = 0$, which are shown in the leftmost images, include fewer blocks and a close proximity between the agent's starting position and the goal. As k increases, environments are generated with a greater number of blocks and a larger distance between the starting position and the goal, resulting in the elimination of all possible paths when $k = M - 1$. The results demonstrate that we can effectively control the difficulty level of the environment using the diffusion-based environment generator and learned environment critic, without domain knowledge. We also present the results of controlling difficulty levels for the 2D bipedal locomotion task in Appendix C.2.

6 Limitation

While the proposed method is suitable for training a robust policy, there exist several limitations. First, despite the existence of the optimal point is proven in Proposition 4.1, convergence to such optimal point is not guaranteed. Furthermore, the difference between the true value of the regret and its estimate is not tightly bounded. Lastly, updating the environment critic using episodic results cannot exactly capture the current agent's capability since the policy is updated after the episode. Hence, exploring methods to estimate the regret with a rigorous theoretical background would be an interesting topic for future work.

7 Conclusion

In this work, we present a novel UED algorithm that exploits the representation power of diffusion models. We first introduce soft UED, which augments the original UED objective with an entropy regularization term to make it suitable for using a diffusion-based environment generator. Then, we propose ADD, which guides the pre-trained diffusion model with a novel regret estimator to produce environments that are conducive to train a robust policy. Our experimental results demonstrate that ADD is capable of training a policy that successfully generalizes to challenging environments. Moreover, it has been verified that ADD generates an instructive curriculum with varying complexity while covering large environment configuration spaces.

Acknowledgments and Disclosure of Funding

This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01190, [SW Star Lab] Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning, 40%), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2022R1A2C2008239, General-Purpose Deep Reinforcement Learning Using Metaverse for Real World Applications, 40%), and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (NO.RS-2021-II211343, Artificial Intelligence Graduate School Program [Seoul National University], 20%).

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.

[2] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan. 2016.

[3] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Proceedings of the Conference on Robot Learning*. PMLR, Dec. 2022.

[4] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, Aug. 2023.

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, Jun. 2019.

[6] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, Jun. 2018.

[7] Nick Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behavior*, 6(2):325–368, Sep. 1997.

[8] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real single-image flight without a single real image. In *Proceedings of Robotics: Sicence and Systems*, Jul. 2017.

[9] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the International Conference on Intelligent Robots and Systems*. IEEE, Sep. 2017.

[10] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Mar. 2020.

[11] 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(1):7382–7431, Jan. 2020.

[12] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, Feb. 2019.

[13] Seungjae Lee, Daesol Cho, Jonghae Park, and H Jin Kim. CQM: Curriculum reinforcement learning with a quantized world model. In *Advances in Neural Information Processing Systems*, Dec. 2024.

[14] 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. In *Advances in Neural Information Processing Systems*, Dec. 2020.

[15] Leonard J Savage. The theory of statistical decision. *Journal of the American Statistical Association*, 46(253):55–67, Mar. 1951.

[16] Abdus Salam Azad, Izzeddin Gur, Jasper Emhoff, Nathaniel Alexis, Aleksandra Faust, Pieter Abbeel, and Ion Stoica. CLUTR: Curriculum learning via unsupervised task representation learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, Jul. 2023.

[17] Dexun Li and Pradeep Varakantham. Enhancing the hierarchical environment design via generative trajectory modeling. *arXiv preprint arXiv:2310.00301*, Feb. 2024.

[18] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *Proceedings of the International Conference on Machine Learning*. PMLR, Jul. 2021.

[19] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In *Advances in Neural Information Processing Systems*, Dec. 2021.

[20] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *Proceedings of the International Conference on Machine Learning*. PMLR, Jul. 2022.

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, Dec. 2020.

[22] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, Dec. 2010.

[23] Peide Huang, Mengdi Xu, Jiacheng Zhu, Laixi Shi, Fei Fang, and Ding Zhao. Curriculum reinforcement learning using optimal transport via gradual domain adaptation. In *Advances in Neural Information Processing Systems*, Dec. 2022.

[24] Daesol Cho, Seungjae Lee, and H Jin Kim. Outcome-directed reinforcement learning by uncertainty & temporal distance-aware curriculum goal generation. In *Proceedings of International Conference on Learning Representations*, Feb. 2023.

[25] Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *Proceedings of the International Conference on Machine Learning*. PMLR, Jul. 2020.

[26] Philip Bontrager and Julian Togelius. Learning to generate levels from nothing. In *Proceedings of the IEEE Conference on Games*, Aug. 2021.

[27] Eric Mazumdar, Lillian J Ratliff, and S Shankar Sastry. On gradient-based learning in continuous games. *SIAM Journal on Mathematics of Data Science*, 2(1):103–131, May 2020.

[28] 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 *Proceedings of the International Conference on Machine Learning*. PMLR, Jul. 2023.

[29] Allen Z Ren and Anirudha Majumdar. Distributionally robust policy learning via adversarial environment generation. *IEEE Robotics and Automation Letters*, 7(2):1379–1386, Apr. 2022.

[30] Zoey Chen, Karl Van Wyk, Yu-Wei Chao, Wei Yang, Arsalan Mousavian, Abhishek Gupta, and Dieter Fox. Learning robust real-world dexterous grasping policies via implicit shape augmentation. In *Proceedings of the Conference on Robot Learning*, Dec. 2022.

[31] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proceedings of the International Conference on Learning Representations*, May 2021.

[32] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Proceedings of the International Conference on Learning Representations*, May 2020.

[33] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in neural information processing systems*, Dec. 2022.

[34] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems*, Dec. 2022.

[35] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the International Conference on Machine Learning*, Jul. 2022.

[36] Mineui Hong, Minjae Kang, and Songhwai Oh. Diffused task-agnostic milestone planner. In *Advances in Neural Information Processing Systems*, Dec. 2023.

[37] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Policy-guided diffusion. *arXiv preprint arXiv:2404.06356*, Apr. 2024.

[38] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, Dec. 2021.

[39] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, Jul. 2022.

[40] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, Jun. 2022.

[41] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, Apr. 2022.

[42] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2023.

[43] Xinquan Chen, Xitong Gao, Juanjuan Zhao, Kejiang Ye, and Cheng-Zhong Xu. Advdiffuser: Natural adversarial example synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Oct. 2023.

[44] Chejian Xu, Ding Zhao, Alberto Sangiovanni-Vincentelli, and Bo Li. Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles. In *Proceedings of the International Conference on Machine Learning Workshop on New Frontiers in Adversarial Machine Learning*, Jul. 2023.

[45] TaeHo Yoon, Kibeom Myoung, Keon Lee, Jaewoong Cho, Albert No, and Ernest K Ryu. Censored sampling of diffusion models using 3 minutes of human feedback. In *Advances in Neural Information Processing Systems*, Dec. 2023.

[46] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, May 1982.

[47] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, Dec. 2020.

[48] Ishita Mediratta, Minqi Jiang, Jack Parker-Holder, Michael Dennis, Eugene Vinitsky, and Tim Rocktäschel. Stabilizing unsupervised environment design with a learned adversary. In *Proceedings of the Conference on Lifelong Learning Agents*. PMLR, Aug. 2023.

[49] Varun Bhatt, Bryon Tjanaka, Matthew Fontaine, and Stefanos Nikolaidis. Deep surrogate assisted generation of environments. In *Advances in Neural Information Processing Systems*, Dec. 2022.

[50] 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. Dec. 2023.

[51] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, Aug. 2017.

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

[53] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and J Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems*, Dec. 2023.

[54] Samuel Garcin, James Doran, Shangmin Guo, Christopher G Lucas, and Stefano V Albrecht. Dred: Zero-shot transfer in reinforcement learning via data-regularised environment design. In *Proceedings of the International Conference on Machine Learning*. PMLR, Jul. 2024.

[55] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, Aug. 2008.

[56] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv:1606.01540*, Jun. 2016.

[57] Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *Proceedings of the International Conference on Machine Learning*, Jul. 2008.

[58] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, Dec. 2016.

[59] Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171–176, Mar. 1958.

[60] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the Medical Image Computing and Computer-Assisted Intervention*, Nov. 2015.

A Algorithm Details

A.1 Proof of Proposition 4.1

In this section, we show that the minimax problem of soft UED (7) has zero minimax duality gap. We assume that S, A, and Θ are finite to avoid the technical issues regarding compactness of a set of distributions. Following Section 3.1, we denote a reward function, transition probability, and initial state distribution of an environment $\mathcal{M}^\theta$ with $\mathcal{R}^\theta$, $\mathcal{P}^\theta$, and ρ_0^θ, respectively.

We first define an occupancy measure, for a policy $\pi \in \Pi$ and an environment $\mathcal{M}^\theta$, as $\rho_\pi^\theta(s,a) = \pi(a|s)\sum_{n=0}^{N}\gamma^n \Pr(s_n = s|\pi,\theta)$. Then, there is a one-to-one correspondence between Π and a set of valid occupancy measures $\mathcal{D}^\theta := \{\rho : \sum_a \rho(s,a) = \rho_0^\theta(s) + \gamma\sum_{s',a'}\mathcal{P}^\theta(s|s',a')\rho(s',a')\}$ [57, 58].

If we define a global occupancy measure as $\rho_\pi := \left[\rho_\pi^{\theta^i}\right]_{i=1}^{|\Theta|}$, where θ^i is an ith element of Θ, it is obvious that there is a one-to-one correspondence between Π and a set of valid global occupancy measures $\mathcal{D} \subset \mathcal{D}^{\theta^1} \times \cdots \mathcal{D}^{\theta^{|\Theta|}}$. Therefore, we can replace π in the objective function of soft UED (7) with ρ_π, as in the following lemma:

Lemma A.1. *if* $\bar{L}(\rho_\pi,\Lambda) = \sum_\theta(V^*(\theta) - \sum_{s,a}\rho_\pi^\theta(s,a)\mathcal{R}^\theta(s,a))\Lambda(\theta) + \frac{1}{\omega}H(\Lambda)$, *where* $V^*(\theta) = \max_{\pi^A\in\Pi} V(\pi^A,\theta)$, *then* $\bar{L}(\rho_\pi,\Lambda) = L(\pi,\Lambda)$.

Proof. Based on the definition of the regret (1), we have

$$\begin{aligned}
L(\pi,\Lambda) &= \mathop{\mathbb{E}}_{\theta\sim\Lambda}[\textsc{Regret}(\pi,\theta)] + \frac{1}{\omega}H(\Lambda) \\
&= \sum_\theta \textsc{Regret}(\pi,\theta)\Lambda(\theta) + \frac{1}{\omega}H(\Lambda) \\
&= \sum_\theta (V^*(\theta) - V(\pi,\theta))\Lambda(\theta) + \frac{1}{\omega}H(\Lambda) \\
&= \sum_\theta (V^*(\theta) - \sum_{s,a}\rho_\pi^\theta(s,a)\mathcal{R}^\theta(s,a))\Lambda(\theta) + \frac{1}{\omega}H(\Lambda) \\
&= \bar{L}(\rho_\pi,\Lambda).
\end{aligned} \tag{14}$$

□

Now, we can rewrite the objective function of soft UED with a global occupancy measure as follows:

$$\min_{\rho\in\mathcal{D}}\max_{\Lambda\in\mathcal{D}_\Lambda}\bar{L}(\rho,\Lambda). \tag{15}$$

Then, we can prove Proposition 4.1 by showing (15) has zero duality gap. However, we cannot apply minimax theorem [59] directly since $\mathcal{D}$ is not a convex set. To resolve this issue, we first augment the problem as follows:

$$\min_{\rho\in\bar{\mathcal{D}}}\max_{\Lambda\in\mathcal{D}_\Lambda}\bar{L}(\rho,\Lambda), \tag{16}$$

where $\bar{\mathcal{D}} := \{\sum_{k=1}^K w_k\rho_k | K\in\mathbb{N}, \sum_{k=1}^K w_k = 1, \forall k\in\{1,\dots,K\} : w_k \geqq 0, \rho_k\in\mathcal{D}\}$ is a convex hull of $\mathcal{D}$. We will show that the augmented problem (16) has zero minimax duality gap, and end the proof by showing the optimal values of the augmented problem can also be reached by the original problem (15).

Lemma A.2. $\min_{\rho\in\bar{\mathcal{D}}}\max_{\Lambda\in\mathcal{D}_\Lambda}\bar{L}(\rho,\Lambda) = \max_{\Lambda\in\mathcal{D}_\Lambda}\min_{\rho\in\bar{\mathcal{D}}}\bar{L}(\rho,\Lambda)$

Proof. Since $\bar{L}(\rho,\Lambda)$ is a linear combination of ρ^θ, it is convex for all ρ. Furthermore, $\bar{L}(\rho,\Lambda)$ is concave for Λ since the entropy H is concave. Therefore, based on the fact that $\bar{\mathcal{D}}$ and $\mathcal{D}_\Lambda$ are both convex and compact, the augmented problem has zero duality gap due to minimax theorem [59]. □

Lemma A.3. *For every $\Lambda \in \mathcal{D}_\Lambda$ and corresponding $\rho^*(\Lambda) \in \operatorname{argmin}_{\rho \in \bar{\mathcal{D}}} \bar{L}(\rho, \Lambda)$, there exists $\rho' \in \mathcal{D}$ such that $\bar{L}(\rho^*(\Lambda), \Lambda) = \bar{L}(\rho', \Lambda)$.*

Proof. For every $\Lambda \in \mathcal{D}_\Lambda$ and corresponding $\rho^*(\Lambda) \in \operatorname{argmin}_{\rho \in \bar{\mathcal{D}}} \bar{L}(\rho, \Lambda)$, there exist $K \in \mathbb{N}, w_{1:K} \geqq 0$, and $\rho_{1:K} \in \mathcal{D}$ such that $\sum_{k=1}^K w_k = 1$ and $\rho^*(\Lambda) = \sum_{k=1}^K w_k \rho_k$. Then, following inequality holds:

$$\min\{\bar{L}(\rho_k, \Lambda)\}_{k=1}^K \geqq \bar{L}(\rho^*(\Lambda), \Lambda) = \sum_{k=1}^K w_k \bar{L}(\rho_k, \Lambda) \geqq \min\{\bar{L}(\rho_k, \Lambda)\}_{k=1}^K, \tag{17}$$

where first inequality holds due to the definition of $\rho^*(\Lambda)$, equality holds since $\bar{L}$ is linear for ρ, and second inequality holds since $\rho^*(\Lambda)$ is a convex combination of $\rho_{1:K}$. Then, $\rho' \in \operatorname{argmin}_{\rho \in \{\rho_k\}_{k=1}^K} \bar{L}(\rho, \Lambda)$ is an element of $\mathcal{D}$ and satisfies $\bar{L}(\rho^*(\Lambda), \Lambda) = \bar{L}(\rho', \Lambda)$. □

Lemma A.3 ensures the minimum value of $\bar{L}$ achieved over $\bar{\mathcal{D}}$ is also achievable over $\mathcal{D}$, implying the optimal value is the same for both the original and augmented problems. It confirms that strong duality holds for the original problem (15) as well.

Proposition A.4. $\min_{\rho \in \mathcal{D}} \max_{\Lambda \in \mathcal{D}_\Lambda} \bar{L}(\rho, \Lambda) = \max_{\Lambda \in \mathcal{D}_\Lambda} \min_{\rho \in \mathcal{D}} \bar{L}(\rho, \Lambda)$

Hence, using Lemma A.1 and Proposition A.4, we can prove Proposition 4.1:

$$\min_{\pi \in \Pi} \max_{\Lambda \in \mathcal{D}_\Lambda} L(\pi, \Lambda) = \min_{\rho \in \mathcal{D}} \max_{\Lambda \in \mathcal{D}_\Lambda} \bar{L}(\rho, \Lambda) = \max_{\Lambda \in \mathcal{D}_\Lambda} \min_{\rho \in \mathcal{D}} \bar{L}(\rho, \Lambda) = \max_{\Lambda \in \mathcal{D}_\Lambda} \min_{\pi \in \Pi} L(\pi, \Lambda) \tag{18}$$

A.2 Diffusion Models

In this section, we present implementation details on diffusion models. To solve the forward and reverse processes (3, 4), we follow the implementation of DDPM [21], which can be viewed as a discretization of VP SDE (3). As as result, the forward process (3) is implemented as follows:

$$\theta_t = \sqrt{1 - \beta_t}\theta_{t-1} + \sqrt{\beta_t} Z_t, \tag{19}$$

where $Z_t \sim \mathcal{N}(0, I)$. In the appendix, we make a slight abuse of notation by considering t as a discrete variable to provide detailed implementation specifics. To solve the reverse process (4), we utilize a error network $\epsilon_\phi(\theta_t, t) = -\sqrt{1 - \alpha_t} s_\phi(\theta_t, t)$ instead of using s_ϕ, where $\alpha_t = \prod_{t'=1}^t 1 - \beta_{t'}$, and β_t follows the linear noise schedule. The error network is trained to minimize the loss $J(\theta_0, t; \phi) := \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} \|\epsilon - \epsilon_\phi(\sqrt{\alpha_t}\theta_0 + \sqrt{1 - \alpha_t}\epsilon)\|^2$. To futher accelerate the sampling, we apply DDIM sampling [32] as follows:

$$\theta_{t-1} = \frac{1}{\sqrt{1 - \beta_t}}\theta_t - \left(\frac{\sqrt{1 - \alpha_t}}{\sqrt{1 - \beta_t}} - \sqrt{1 - \frac{\alpha_t}{1 - \beta_t}}\right)\epsilon_\phi(\theta_t, t). \tag{20}$$

Since DDIM sampling (20) is deterministic, we can sample the environment parameter θ with T' denoising steps, which is less than the original diffusion timestep T.

A.3 Environment Critic Update

In this section, we present details on updating the environment critic. After the interaction between the RL agent with a policy π and an environment $\mathcal{M}_\theta$, we construct a target return distribution $\mathcal{Z}_\pi^{\text{target}}(\theta)$ using episodic returns. Specifically, if episodic returns that the RL agent achieves are $\{v_k\}_{k=0}^{K-1}$, we give equal probabilities to each return and project them into the support $\{z_i\}_{i=0}^{M-1}$. As a result, the target return distribution is constructed as follows:

$$\mathcal{Z}_\pi^{\text{target}}(\theta) = z_i \quad \text{w.p.} \quad \frac{1}{K} \sum_{k=0}^{K-1} \left[1 - \frac{|v_k - z_i|}{\Delta}\right]_0^1, \tag{21}$$

where $[\cdot]_0^1$ bounds its argument in the range $[0, 1]$, and $\Delta := \frac{(v_{max} - v_{min})}{M}$ is a width of each bin. Then, we train the environment critic to produce the return distribution close to the target distribution

(21). To that end, we first sample t from $1, \ldots, T$ and obtain θ_t by solving the forward process (3) with initial condition $\theta_0 = \theta$, then construct the estimated return distribution $\hat{\mathcal{Z}}_\pi(\theta_t, t)$ using an output of the environment critic $\tau_\psi(\theta_t, t)$. Then, ψ is updated via gradient descent to minimize the cross entropy loss between $\hat{\mathcal{Z}}_\pi(\theta_t, t)$ and $\mathcal{Z}_\pi^{\text{target}}(\theta)$. To prevent overfitting, we store episodic results in a buffer, then sample the environment parameters and their corresponding target return distribution from the buffer for training the environment critic.

A.4 Pseudocode of ADD

Algorithm 1 Adversarial Environment Design via Regret-Guided Diffusion Models

Input: Policy network π_ξ, diffusion model s_ϕ, and environment critic network τ_ψ.
Initialize network parameters ξ, ϕ, ψ.
Train the diffusion model s_ϕ on a dataset of randomly generated environments.
for each epochs **do**
  Sample a set of environment parameters $\{\theta^i\}_{i=1}^B$ via regret-guided reverse process (10), whose regret is estimated using the environment critic (12).
  Run episode on a set of environments $\{\mathcal{M}^{\theta^i}\}_{i=1}^B$ and update the policy π_ξ via RL.
  Update the environment critic τ_ψ using episodic returns (Appendix A.3).
end for

B Experiment Details and Hyperparameters

In this section, we provide a comprehensive explanation of the experiments discussed in Section 5. We begin by detailing the two tasks: partially observable navigation and 2D bipedal locomotion. Then, we conclude the section by reporting the hyperparameters employed in the experiments.

B.1 Partially Observable Navigation Task

Environment details. In the partially observable navigation task, which is based on the Minigrid [53] and adopted for UED in prior works [14], the agent is trained to find and reach a goal in the grid maze environment. Each maze environment is a 15×15 grid whose cells on the edge are all walls, and the cells inside can contain walls, agents, or goals. When the agent reaches the goal, it receives a reward of $1 - N/N_{max}$, where N is a length of the episode and N_{max} is a maximum length of each episode. If it does not reach the goal, it receives a reward of 0. The agent uses a 7×7 grid around itself and its direction as an observation and chooses one of the actions: turn left, turn right, or go forward.

Environment generation. Aligning with Parker-Holder et al. [20], we limit the number of walls that can exist in the 13 x 13 grid, excluding the walls on the edges, to 60. The RL generator of PAIRED selects locations to place walls over 60 steps, ensuring no changes if a wall already exists. After placing all the walls, it selects the starting position of the agent and the goal location. If a wall exists at those locations, it removes the wall and places the agent or the goal. On the other hand, the random generator used by PLR$^\perp$ and ACCEL uniformly samples the number of walls between 0 and 60 in advance and then choose the position to place the walls, agent and goal location randomly. This random generation is also used to create a dataset for training the diffusion-based environment generator of the proposed algorithm. Specifically, each environment parameter data is represented with a $13 \times 13 \times 3$ image. The first channel represents the location of the walls, with a value of one if a wall is present in the cell, and zero if it is not. The second channel indicates the starting position of the agent, with a value of one for the starting cell, 0.5 for the cell after moving forward once, and zero for all other cells. Finally, the third channel has a value of one for the cell corresponding to the goal, and zero otherwise. After training the diffusion-based environment generator on the randomly generated dataset, we can produce the environment parameter by employing the trained generator to solve reverse process (5), as shown in Figure 5.

Training time. All methods are trained utilizing RTX 3090Ti. To train DR and ADD w/o guidance, they require almost 48 hours to run 250 million environment steps. ADD requires almost 56 hours, and PLR$^\perp$ and ACCEL requires almost 100 hours for each random seed.

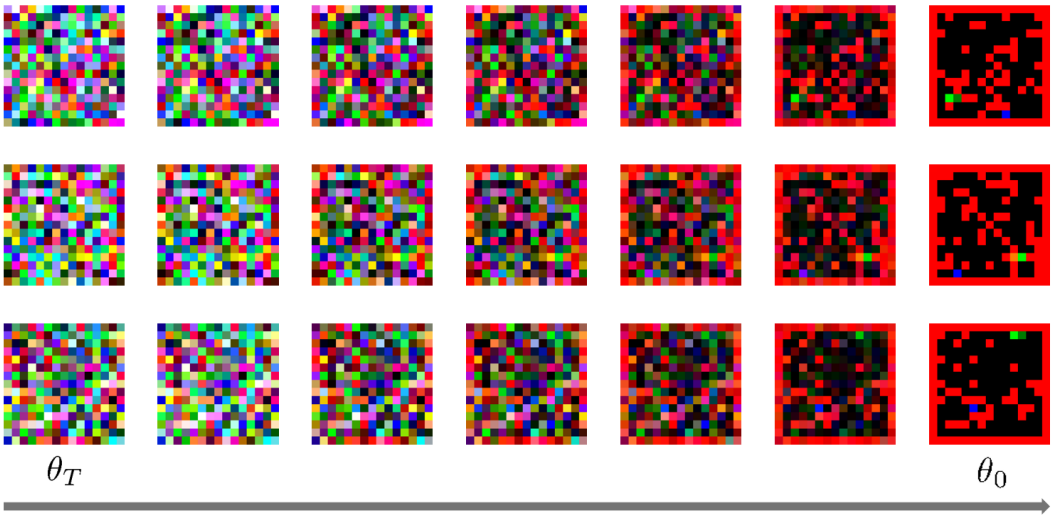

Figure 5: **Maze environment generation using diffusion models.** We represent the maze environment with a parameter $\theta \in \mathbb{R}^{13\times13\times3}$, with each channel indicating the location of walls, the agent, and the goal. After training the diffusion-based environment generator on a dataset of randomly generated environment parameters, we can sample maze environments by solving the reverse process (5).

B.2 2D Bipedal Locomotion Task

Environment details. For the 2D bipedal locomotion task, we conduct the experiment using a modified version of the BipedalWalker environment from OpenAI Gym [56], as done in Parker-Holder et al. [20]. In this task, the agent is trained to walk over challenging terrains by controlling four joints, and action is decided using a 24-dimensional observation, which is consisting of Lidar measurements, linear and angular velocities of the robot, positions and speeds of joints, and contact information. The agent receives a positive reward for moving forward, and receives -100 as a reward if it falls to the ground. If the agent reaches the opposite end of the terrain, total reward it receives is over 300.

Environment generation. In the 2D bipedal locomotion task, we generate the environment by deciding the eight-dimensional environment parameter $\theta \in \mathbb{R}^8$, which is consisting of a min / max stump height, min / max stair height, min / max pit gap , stair steps, and roughness of the terrain. The RL generator of PAIRED selects each parameter sequentially, and the random generator of PLR$^\perp$ and ACCEL randomly decide each parameter by sampling a real number from its domain, which is reported in Table 1. We employ the random generator to construct a dataset of environment parameters, and train the diffusion-based environment generator, which will be used to produce the environment parameter while training the agent using the proposed method. After the environment parameter is decided, the entire environment is generated by the procedural content generation algorithm.

Training time. All methods are trained utilizing RTX 3090Ti. To train DR and ADD w/o guidance, they require almost 80 hours to run 2 billion environment steps. ADD requires almost 92 hours, and PLR$^\perp$ and ACCEL requires almost 160 hours to run the experiment for each random seed.

Table 1: **Domain of the environment parameter for the 2D bipedal locomotion task.**

	Stump Height	**Stair Height**	**Pit Gap**	**Stair Steps**	**Roughness**
Domain	$[0.0, 5.0]$	$[0.0, 5.0]$	$[0.0, 10.0]$	$[0.0, 10.0]$	$[1.0, 9.0]$

B.3 Hyperparameters

To train RL agents using UED baselines, we follow the implementation and hyperparameters of Parker-Holder et al. [20], which is available at https://github.com/facebookresearch/dcd. The same parameters and network architecture were used to train the ADD agent, and the hyperparameters used are reported in the Table 2. To train the diffusion-based environment generator, we followed the implementation of Dhariwal et al. [38] and Yoon et al. [45], which is available at https://github.com/tetrzim/diffusion-human-feedback, and the size of the randomly generated dataset is set to 10 million. For the partially observable navigation task, the architecture of the error network ϵ_ϕ follows the UNet [60], which utilizes three residual blocks with channel multipliers $[1, 2, 2, 2]$ for each resolution. For the 2D bipedal locomotion task, the architecture of error network is four-layer MLP with a sinusoidal time embedding. We report detailed hyperparameters used to train diffusion models in Table 3. Lastly, we report detailed hyperparameters for regret-guided diffusion process and training the environment critic in Table 4. The network architecture of the environment critic is based on the UNet encoder for the partially observable navigation task and a four-layer MLP for the 2D bipedal locomotion task.

Table 2: **Hyperparameters used for training the RL agent in each task**

Parameter	**Minigrid**	**BipedalWalker**
γ	0.995	0.99
λ_{GAE}	0.95	0.9
PPO rollout length	256	2000
PPO epochs	5	5
PPO minibatches for epoch	1	32
PPO clip range	0.2	0.2
PPO number of workers	32	16
Adam learning rate	1e-4	2e-4
Adam ϵ	1e-5	1e-5
PPO max gradient norm	0.5	0.5
PPO value clipping	True	False
Return normalization	False	True
Value loss coefficient	0.5	0.5
Entropy coefficient	0.0	1e-3
LSTM-based policy	True	False

Table 3: **Hyperparameters used for training the diffusion-based environment generator**

Parameter	**Minigrid**	**BipedalWalker**
DDPM timestep T	1000	1000
Network architecture	UNet	MLP
hidden dimension	128	256
Batch size	128	512
Dropout	0.0	0.0
AdamW learning rate	1e-4	1e-4
AdamW weight decay	0.05	0.0
AdamW β_1	0.9	0.9
AdamW β_2	0.999	0.999
EMA rate	0.9999	0.9999
Number of training steps	3e5	1.5e5

Table 4: **Hyperparameters used for the regret guidance and training the environment critic**

Parameter	Minigrid	BipedalWalker
DDIM timestep T'	50	200
Number of bins M	100	100
Return domain $[v_{min}, v_{max}]$	$[0, 1]$	$[0, 300]$
Guidance weight ω	5.0	15.0
CVaR risk level α	0.15	0.3
Environment critic minibatches	128	128
Environment ciritic epochs	5	5
Environment critic buffer size	1600	800

C Detailed Experimental Results

In this section, we present detailed experimental results in the partially observable navigation task and 2D bipedal locomotion task. For each task, we will provide specific zero-shot generalization performance, t-SNE plots of all baselines, and the training environments generated by the proposed algorithm. Additionally, we will show examples of environments generated with varying difficulty levels using the method described in Section 4.4.

C.1 Partially Observable Navigation Task

Zero-shot transfer test results. After training the RL agent in the partially observable navigation task, we evaluate the generalization capability of the learned policy by testing the agent in twelve unseen environments, which are shown in Figure 6. In each environment, the agent is evaluated for 100 independent episodes, and the full result is reported in Table 5. We note that the reported quartile values represent the average of the quartile values of the solved rates achieved in the test environments for each seed. The results demonstrate that the proposed method achieves the best performance in five out of 12 test environments, with particularly high mean and quartile values compared to the baselines. Therefore, we can infer that ADD successfully trains an agent that is robust to environmental changes and generalizes to various environments.

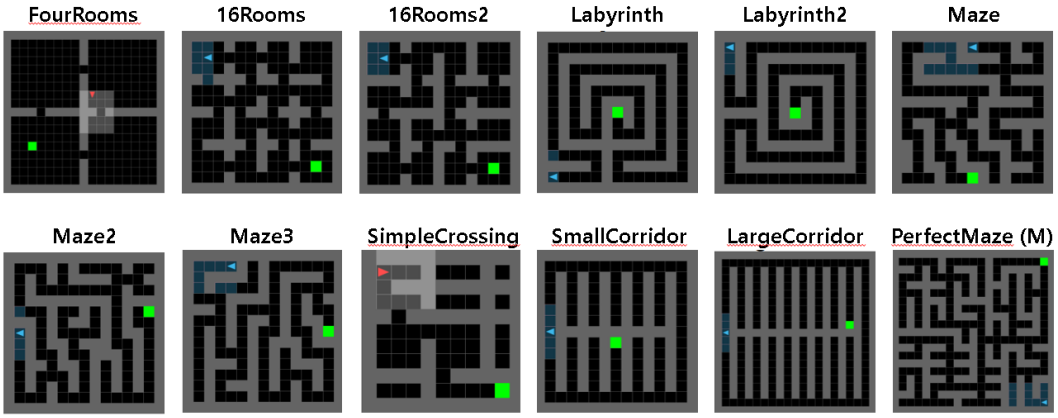

Figure 6: **Zero-shot test environments for the partially observable navigation task.** SimpleCrossing and FourRooms environments are adopted from Chevalier-Boisvert et al. [53], and other test environments are adopted from Dennis et al. [14] and Jiang et al. [19, 20].

Table 5: **Partially observable navigation task results.** The table shows the average solved rate and standard deviation over five independent runs, and each run is evaluated by 100 independent episodes for each test environment. All methods train the agent using LSTM-based PPO and evaluated after 250M environmental steps.

Environment	DR	PAIRED	PLR$^{\perp}$	ACCEL	ADD w/o guidance	ADD
FourRooms	0.62 ± 0.01	0.51 ± 0.05	**0.64 ± 0.10**	0.51 ± 0.05	0.61 ± 0.07	0.61 ± 0.09
16Rooms	0.78 ± 0.21	0.96 ± 0.41	0.72 ± 0.09	**0.97 ± 0.05**	0.93 ± 0.15	0.91 ± 0.13
16Rooms2	0.50 ± 0.36	0.47 ± 0.28	0.60 ± 0.50	0.59 ± 0.41	0.63 ± 0.39	**1.00 ± 0.10**
Labyrinth	0.74 ± 0.42	0.74 ± 0.47	0.63 ± 0.42	0.96 ± 0.08	0.56 ± 0.41	**1.00 ± 0.00**
Labyrinth2	0.62 ± 0.49	0.49 ± 0.46	0.64 ± 0.50	0.73 ± 0.36	0.51 ± 0.54	**0.97 ± 0.04**
Maze	0.36 ± 0.47	0.06 ± 0.39	0.21 ± 0.08	**0.82 ± 0.32**	0.18 ± 0.31	0.79 ± 0.37
Maze2	0.46 ± 0.48	0.60 ± 0.17	0.18 ± 0.54	**0.97 ± 0.03**	0.62 ± 0.44	0.76 ± 0.42
Maze3	**0.96** ± 0.09	0.64 ± 0.18	0.90 ± 0.42	0.61 ± 0.48	0.77 ± 0.38	0.74 ± 0.31
SimpleCrossing	0.87 ± 0.06	0.82 ± 0.04	**0.88 ± 0.09**	0.75 ± 0.10	0.87 ± 0.02	0.80 ± 0.13
SmallCorridor	0.63 ± 0.31	0.79 ± 0.02	**0.97 ± 0.18**	0.55 ± 0.46	0.84 ± 0.23	0.94 ± 0.09
LargeCorridor	0.74 ± 0.35	0.43 ± 0.23	0.79 ± 0.36	0.56 ± 0.51	0.65 ± 0.33	**0.95 ± 0.05**
PerfectMaze (M)	0.45 ± 0.20	0.43 ± 0.13	0.37 ± 0.24	0.64 ± 0.19	0.45 ± 0.21	**0.82 ± 0.18**
Mean	0.64 ± 0.13	0.59 ± 0.10	0.63 ± 0.07	0.72 ± 0.07	0.63 ± 0.20	**0.85 ± 0.05**
First quartile	0.39 ± 0.25	0.26 ± 0.25	0.32 ± 0.17	0.55 ± 0.15	0.41 ± 0.28	**0.79 ± 0.07**
Second quartile	0.76 ± 0.19	0.62 ± 0.16	0.71 ± 0.13	0.85 ± 0.05	0.66 ± 0.30	**0.93 ± 0.05**
Third quartile	0.89 ± 0.09	0.92 ± 0.08	0.95 ± 0.01	0.98 ± 0.01	0.86 ± 0.17	**0.99 ± 0.01**

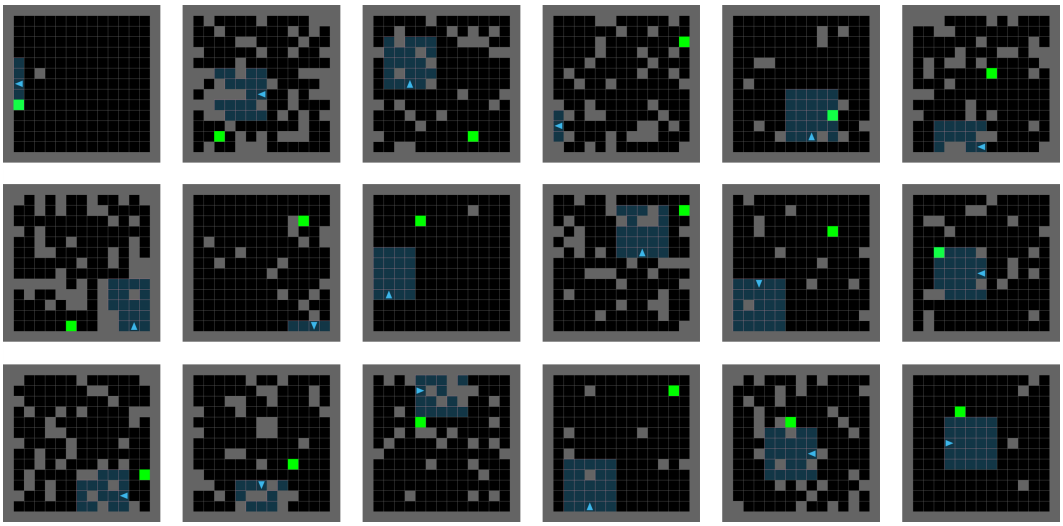

Figure 7: **Examples of environments generated at the beginning of the agent's training in the partially observable task**. The figure shows 18 example environments that are generated right after the initiation of the agent's learning.

Generated training environments. To support the claim regarding the generated curriculum, we provide example scenes of generated training environments in Figure 7 and Figure 8. Comparing two figures reveals that the environments generated after training the agent with 200 million environmental steps are more complex and contain a larger number of blocks, aligning with the quantitative results shown in Figure 2(c). The reason for this difference is that as the agent is trained, it achieves near-optimal performance in simple environments. Consequently, the environment critic predicts that the agent's regret will be larger in environments with a greater number of blocks and increased complexity.

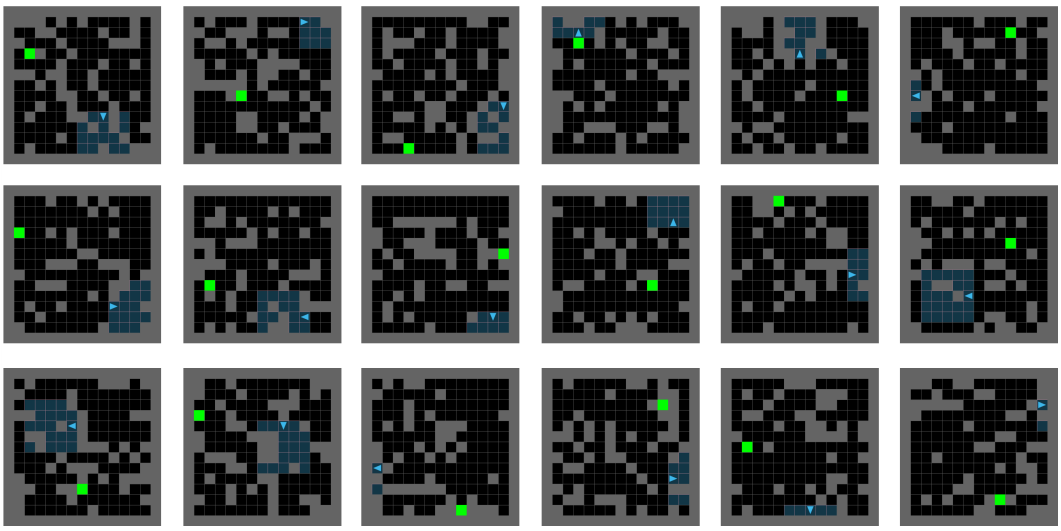

Figure 8: **Examples of environments generated after 200 million environmental steps in the partially observable task**. The figure shows 18 example environments that are generated after 200 million environmental steps

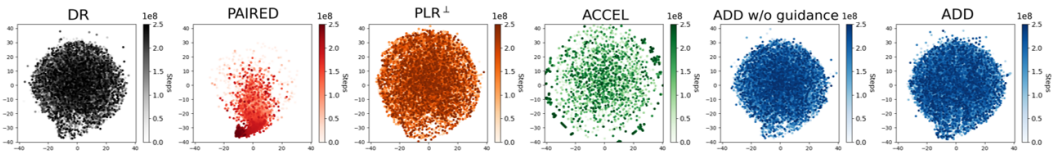

Figure 9: **t-SNE plots of training environments in the partially observable navigation task**. The figure shows t-SNE plots of ADD and baselines. The results are obtained by mapping the environment parameters to a latent space using the encoder of the learned environment critic, followed by training a t-SNE.

t-SNE plots. To conduct a qualitative analysis on the generated environments, we visualize training environments generated from ADD and baselines. Since the parameter of the maze environment is high-dimensional and hard to define the meaningful distance, we first utilize the encoder of the environment critic to map environment parameters to the learned latent space, then train a t-SNE on the latent vectors. The t-SNE results are shown in Figure 9. The results demonstrate that the proposed method generate sufficiently diverse environments comparable to those produced by a random generator.

C.2 2D Bipedal Locomotion Task

Zero-shot transfer test results. For the 2D bipedal locomotion task, we assess the zero-shot performance of the trained RL agent in six unseen test environments shown in Figure 10. In each test environment, we evaluate the performance for 100 independent episodes. In this task, PAIRED struggles to train the RL-based environment generator since the value function often diverges. Therefore, we adopted the zero-shot performance result of PAIRED reported in Parker Holder et al. [20]. We report the full results in Table 6, and the results demonstrate that the proposed method outperforms baselines across all test environments.

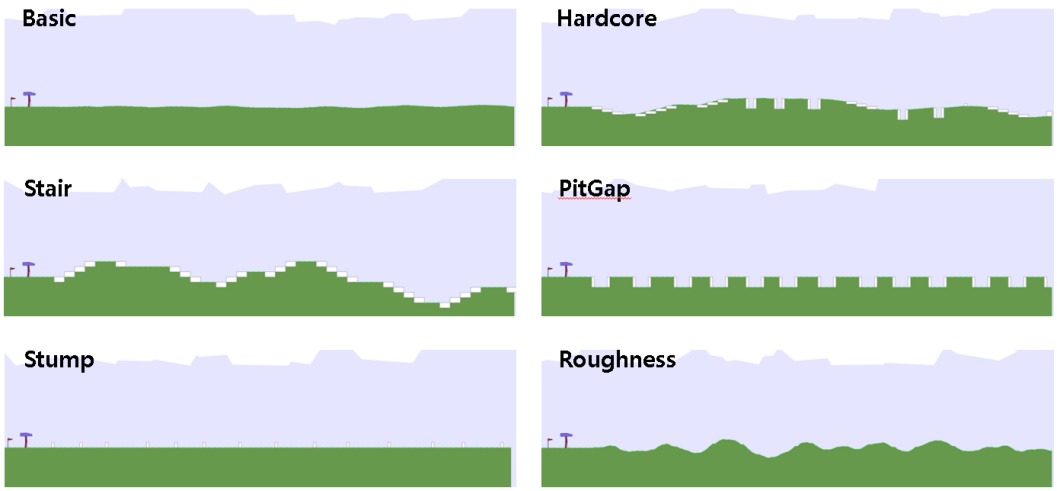

Figure 10: **Test environments for the 2D bipedal locomotion task**. Basic and Hardcore environments are adopted from OpenAI Gym [56], and other four test environments are adopted from Parker Holder et al. [20].

Table 6: **2D bipedal locomotion task results.** The table shows the average return and standard deviation over five independent runs, and each run is evaluated by 100 independent trials on each test environment. All methods train the agent using PPO and evaluated after two billion environmental steps.

Environment	DR	PAIRED	PLR$^{\perp}$	ACCEL	ADD w/o guidance	ADD
Basic	153.7 ± 75.5	206.5 ± 30.3	306.0 ± 5.6	252.0 ± 21.9	280.8 ± 27.3	**312.0 ± 2.5**
Hardcore	10.6 ± 10.1	-47.2 ± 10.6	116.6 ± 27.8	53.1 ± 26.6	35.5 ± 16.5	**140.1 ± 28.8**
Stairs	10.5 ± 8.1	-27.4 ± 12.1	58.4 ± 23.6	33.6 ± 18.5	24.3 ± 9.4	**75.4 ± 34.1**
PitGap	-1.8 ± 10.7	-68.2 ± 9.7	54.2 ± 6.8	29.6 ± 22.4	6.7 ± 20.4	**143.2 ± 74.9**
Stump	-18.5 ± 20.9	-76.0 ± 10.3	9.2 ± 26.3	-23.2 ± 20.8	-16.3 ± 6.3	**58.2 ± 52.6**
Roughness	22.5 ± 12.5	-5.1 ± 25.9	144.5 ± 30.8	85.2 ± 21.8	85.1 ± 26.6	**168.9 ± 38.8**
Mean	29.5 ± 16.5	-2.9 ± 14.5	114.8 ± 17.3	71.7 ± 15.9	69.3 ± 12.0	**149.6 ± 33.0**

Generated training environments. To demonstrate the generated environments vary as the agent learns, we provide the generated environments in the early and later stages of the training in Figure 11 and Figure 12. It can be observed that the environments shown in Figure 12 are much simpler compared to those in Figure 11, which is consistent with the quantitative analysis provided in Figure 3(b). This difference occurs because randomly sampling environment parameters can result in overly challenging environments that hinder the agent's learning, and the environment critic guides the generator to produce simpler environments that are conducive to the agent's learning.

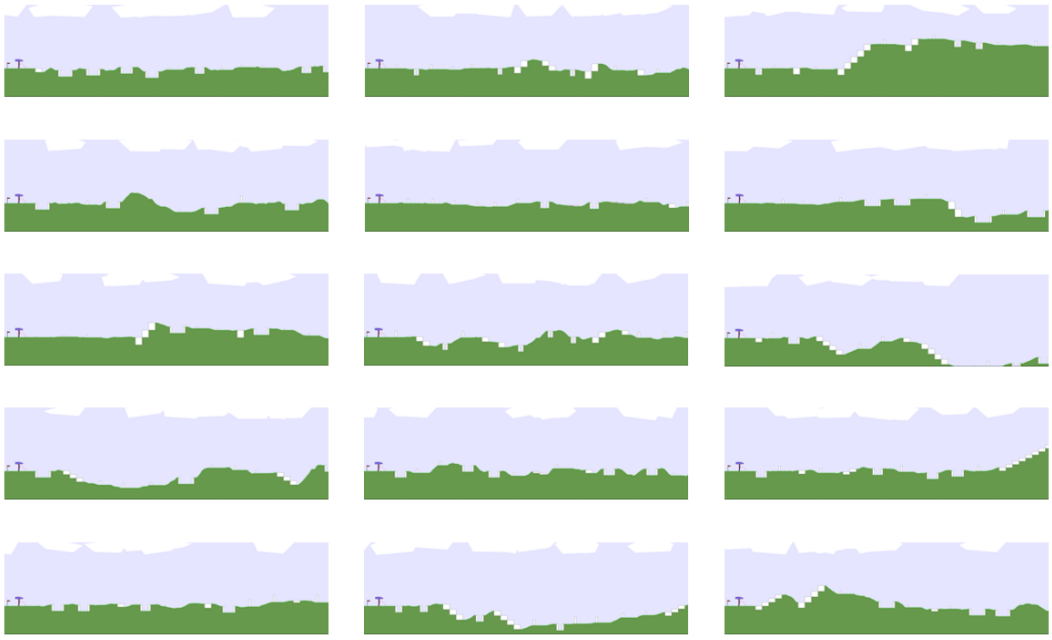

Figure 11: **Examples of environments generated at the beginning of the agent's training in the 2D bipedal locomotion task**. The figure presents 15 example environments that are generated right after the agent starts learning.

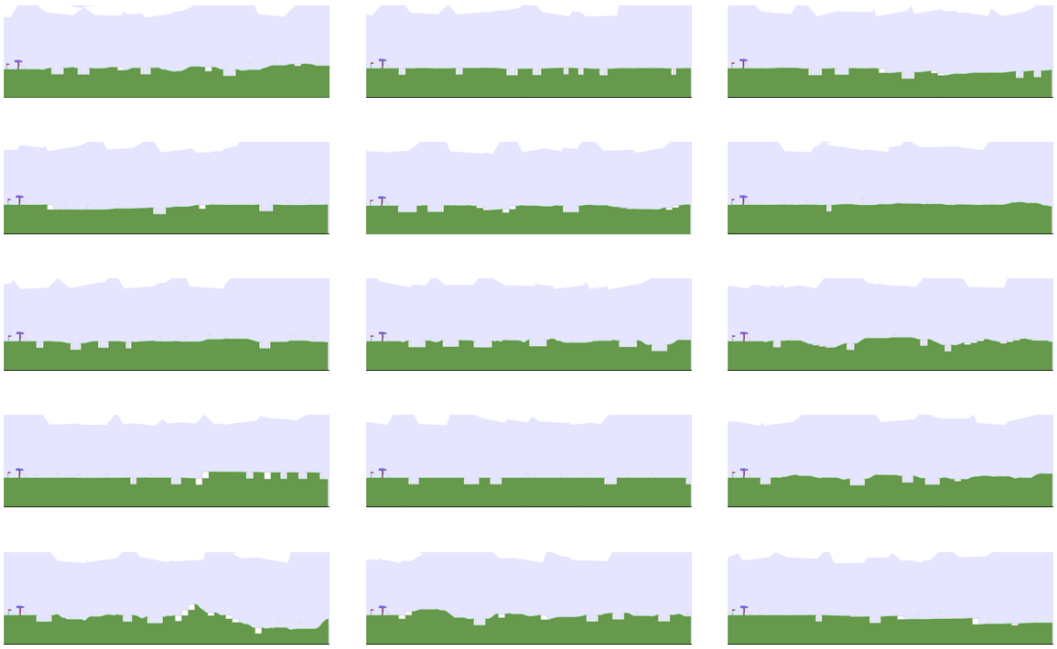

Figure 12: **Examples of environments generated after 1.5 billion environmental steps in the 2D bipedal locomotion task**. The figure presents 15 example environments that are generated after 1,5 billion environmental steps.

t-SNE plots. As in the previous task, we trained a t-SNE to visualize the generated environments for the 2D bipedal locomotion task. Since the environment parameters for this task have low dimensionality and the meaningful distance can be computed using L2 distance, we use the raw environment parameters without mapping them to a learned latent space. The results shown in Figure 13 demonstrate that the proposed method produces sufficiently diverse training environments. In contrast, training environments generated by PAIRED are concentrated in a small area. Due to this difference, the agent trained with the proposed algorithm demonstrates better performance across all test environments.

Controllable generation. As done in the partially observable navigation task, we provide the results of manipulating the difficulty of the generated environments in Figure 14. The results show that when we guide the generator to produce low-difficulty environments, nearly flat terrains are produced. As the desired difficulty increases, the environments become progressively more complex and challenging to walk over. This additional capability of ADD allows the trained environment critic and diffusion-based generator to be reused in applications such as benchmark generation.

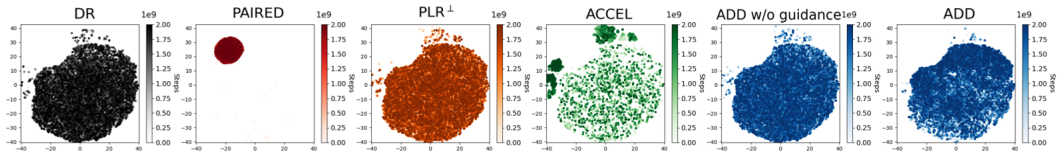

Figure 13: **t-SNE plots of training environments in the 2D bipedal locomotion task**. The figure shows t-SNE plots of ADD and baselines, which are obtained by training a t-SNE to visualize environment parameters in the two-dimensional space.

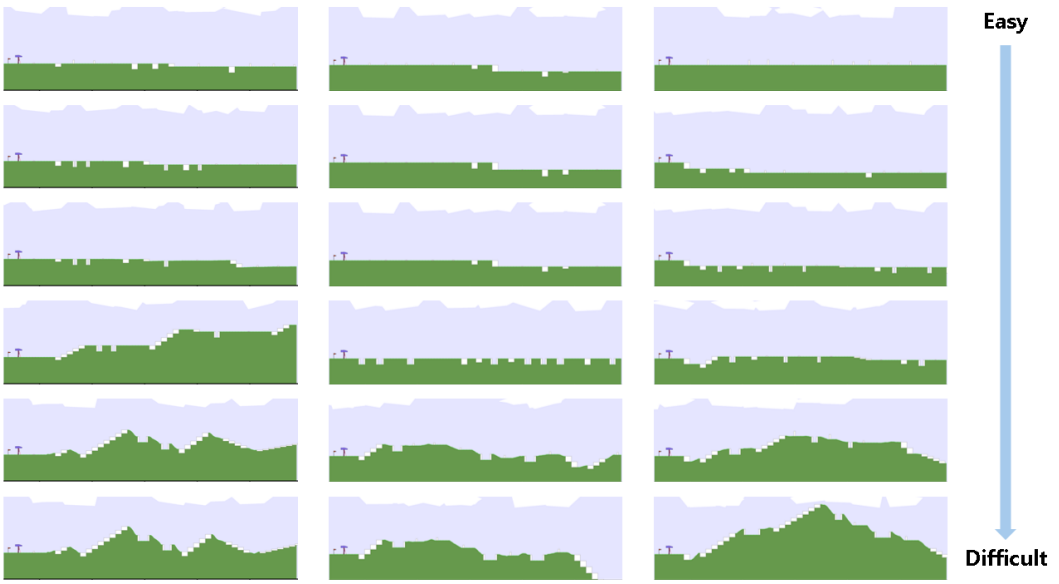

Figure 14: **Controllable generation results in the 2D bipedal locomotion task**. The figure shows how the environments change as the desired difficulty level increases. We note that each column is generated using the same initial noise θ_T and random seed.

C.3 Ablation Study

We conduct ablation studies to analyze the role of entropy regularization term and the number of environments used in diffusion pre-training. Frist, to show whether adding an entropy term to the original UED objective plays a critical role, we measure the zero-shot generalization performance of the trained agent with varying ω, which is defined in the soft UED objective $\mathbb{E}_{\theta\sim\Lambda}\left[\textsc{Regret}(\pi,\theta)\right] + \frac{1}{\omega}H(\Lambda)$. The experimental results are shown in Table 7. From the results, we observed that performance decreases as ω becomes large. Since the influence of the entropy term diminishes as ω increases, it can be seen that our experimental results highlight the importance of the entropy term.

Next, to analyze the influence of the number of samples used during the pre-training phase, we trained the diffusion model using one million samples, which is 100 times fewer than in the original experiment, and measured the performance of the proposed algorithm. The result is a mean success rate of 0.76 ± 0.07 in the partially observable navigation task. This is about 11% lower than the result reported in the original experiment. The result demonstrates that a larger number of samples used in pre-training would lead to better performance, which is quite trivial since the diffusion model can generate more diverse environments when trained with a larger number of samples. Additionally, we note that since we are dealing with an unsupervised setting and the samples used in pre-training are generated through random sampling, there is no need to worry about data scarcity.

Table 7: **Ablation study on the entropy regularization term.** The table shows the zero-shot generalization performance in the partially observable navigation task in accordance to the entropy coefficient ω. We measure the average success rate over five independent seeds.

ω	5	10	20	40	80
Mean success rate	0.85 ± 0.05	0.81 ± 0.05	0.82 ± 0.03	0.64 ± 0.07	0.47 ± 0.16

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions and scope are accurately described in the abstract and instruction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the work are discussed in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the full set of assumptions in Proposition 4.1, and the complete proof is presented in Appendix A.1

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide experiment details and hyperparameters in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We submit the code with a detailed instruction, and all experiments are reproducible.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All the training and test details are provided in Appendix B and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In Figure 2 and Figure 3, all the results are accompanied by error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the type of compute workers and computation time in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research fully conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work is not expected to cause direct societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work does not have a negative use case, so we don't need special safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all the papers that produced the code package, and provide URLs in Appendix B.3.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the details about training in Appendix B.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.