# A HIERARCHICAL APPROACH TO ENVIRONMENT DESIGN WITH GENERATIVE TRAJECTORY MODELING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Unsupervised Environment Design (UED) is a paradigm for training generally capable agents to achieve good zero-shot transfer performance. This paradigm hinges on automatically generating a curriculum of training environments. Leading approaches for UED predominantly use randomly generated environment instances to train the agent. While these methods exhibit good zero-shot transfer performance, they often encounter challenges in effectively exploring large design spaces or leveraging previously discovered underlying structures, To address these challenges, we introduce a novel framework based on Hierarchical MDP (Markov Decision Processes). Our approach includes an upper-level teacher's MDP responsible for training a lower-level MDP student agent, guided by the student's performance. To expedite the learning of the upper leavel MDP, we leverage recent advancements in generative modeling to generate synthetic experience dataset for training the teacher agent. Our algorithm, called *S*ynthetically-enhanced *H*ierarchical *E*nvironment *D*esign (*SHED* ), significantly reduces the resource-intensive interactions between the agent and the environment. To validate the effectiveness of *SHED* , we conduct empirical experiments across various domains, with the goal of developing an efficient and robust agent under limited training resources. Our results show the manifold advantages of *SHED* and highlight its effectiveness as a potent instrument for curriculum-based learning within the UED framework. This work contributes to exploring the next generation of RL agents capable of adeptly handling an ever-expanding range of complex tasks.

## 1 INTRODUCTION

The advances of reinforcement learning (RL Sutton et al. (1998)) have promoted research into the problem of training autonomous agents that are capable of accomplishing complex tasks. One interesting, yet underexplored, area is training agents to perform well in unseen environments, a concept known as zero-shot transfer performance. To this end, Unsupervised Environment Design (UED Dennis et al. (2020)) has emerged as a promising paradigm to address this problem. The primary objective of UED is to automatically generate environments in a curriculum-based manner, and training agents in these sequentially generated environments can equip agents with a general capability, enabling agents to learn robust and adaptive behaviors that can be transferred to new scenarios without explicit exposure during training.

Existing approaches in UED primarily focus on building an adaptive curriculum for the environment generation process to train more generally capable agents. For example, Dennis et al. (2020) formalize the problem of finding adaptive curricula through a game involving an adversarial environment generator (teacher agent), an expert antagonist agent, and the protagonist agent (student agent). The RL teacher is designed to generate environments that maximize regret, defined as the difference between the protagonist and antagonist agent's return. They show that these agents will reach a Nash Equilibrium where the protagonist agent learns the minimax regret policy. However, since teacher agents adapt solely based on regret feedback, it is inherently difficult to adapt to student policy changes, and training such an RL teacher remains a challenge. With the growing popularity of domain randomization (Tobin et al., 2017) due to its promising empirical results, Jiang et al. (2021b) propose randomly generating environments and then curating randomly sampled environments to achieve high regret. Parker-Holder et al. (2022) then propose the adaptive curricula by manually designing a principled, regret-based curriculum, suggesting randomly generating environ-

ment instances with increasing complexity. Li et al. (2023) incorporate diversity measurement when randomly generating new environments to ensure that the agent is exposed to a diverse set of environments. Although these domain randomization-based algorithms achieve good zero-shot transfer performance, they face limitations in efficiently exploring large design spaces and leveraging previously discovered inherent structures within the environments. Moreover, existing UED approaches often rely on open-ended domains, which requires a long training horizon. This is unrealistic in the real world due to limited resources. We aim to design a teacher's policy that generates environments best suited to the current student skill level, thereby achieving an optimal general capability within a finite time horizon setting.

In this paper, we aim to address these limitations by proposing an alternative framework for automatic adaptive environment design. The core idea involves the use of hierarchical Markov Decision Processes (MDPs) to simultaneously formulate the evolution of both the teacher and student during training. This process includes training an upper-level MDP for a teacher agent, which guides the training of the lower-level MDP of a student agent. This guidance is based on a more accurate representation of the student's policy, achieved by evaluating its performance across a diverse set of evaluation environments. This hierarchical approach allows for the optimization of the student policy's capability trajectory through direct training of a teacher policy. However, this method presents a challenge: each transition in the upper-level MDP consists of one/several complete lower-level student MDPs. Consequently, collecting the upper-level teacher agent's experience is slow and resource-intensive.

To accelerate the resource-intensive collection of upper-level MDP experience, we utilize advancements in generative models, such as diffusion models, adapting them to learn a trajectory model. Diffusion models can generate new data points that capture complex distribution properties, such as skewness and multi-modality, exhibited in the collected dataset (Saharia et al., 2022). Specifically, we employ a diffusion probabilistic model (Sohl-Dickstein et al., 2015; Ho et al., 2020) to learn a model of student policy's capability trajectory. Our trained diffusion model is capable of generating new synthetic experiences of student policy's trajectory, which can be used for teacher training. To that end. we introduce *S*ynthetically-enhanced *H*ierarchical *E*nvironment *D*esign (*SHED*), an adaptive approach that can automatically generate increasingly complex environments suited to the current capabilities of the student agent, resulting in promoting continuous learning.

In summary, we make the following contributions:

- We build a novel hierarchical MDPs framework for UED, and provide a straightforward way to represent the student policy.
- We introduce *SHED*, which utilizes diffusion-based techniques to generate synthetic experiences. This method can accelerate the off-policy training process of the teacher agent.
- Our experiments across various domains demonstrate that our methods significantly outperform the existing state-of-the-art method in UED.

We believe that our method has the potential to significantly reduce the time and effort required to design suitable training environments for student agents, enabling the development of more capable and robust RL systems across a wide range of domains under the finite training horizon.

## 2 PRELIMINARIES

In this section, we provide an overview two main research areas on which our work is based.

### 2.1 UNSUPERVISED ENVIRONMENT DESIGN

Dennis et al. (2020) first define UED in terms of an underspecified Partially Observable Markov Decision Process (UPOMDP), which is a tuple $\mathcal{M} =< \mathbb{A}, \mathbb{O}, \theta, \mathbb{S}^{\mathcal{M}}, \mathcal{P}^{\mathcal{M}}, \mathcal{I}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, \gamma >$. The objective is to generate a sequence of environments that effectively support the continual learning of the agent policy. Here $\mathbb{A}$ represents the set of actions, $\mathbb{O}$ is the set of observations, $\mathbb{S}^{\mathcal{M}}$ is the set of states determined by the underspecified environment $\mathcal{M}$, similarly, $\mathcal{P}^{\mathcal{M}}$ is the environment-dependent transition function, and $\mathcal{I}^{\mathcal{M}} : \mathbb{A} \to \mathbb{O}$ is the environment-dependent observation function, $\mathcal{R}^{\mathcal{M}}$ is the reward function, and $\gamma$ is the discount factor. The students is trained to maximize their

cumulative reward $V^{\mathcal{M}}(\pi) = \sum_{t=0}^{T} \gamma^t r_t$ for the current environment under a finite time horizon $T$, and $r_t$ are the collected rewards. Existing works on UED consist of two main strands: the RL-based environment generation approach and the domain randomization-based environment generation approach.

The RL-based generation approach was first formalized by Dennis et al. (2020) as a self-supervised RL paradigm for generating a distribution over environments. This approach involves co-evolving an environment generator policy (teacher) with an agent policy (student), where the teacher's role is to create an environmental distribution that best supports the student agent's continued training. The teacher is trained to produce challenging yet solvable environments that maximize the defined objective of regret, which quantifies the performance difference between the current student agent and a well-trained agent within the current environment.

The domain randomization-based generation approach, on the other hand, involves randomly generating environments. Jiang et al. (2021b) propose to prioritize the storage of environments with high regret values, which are defined using Generalized Advantage Estimation (GAE) (Schulman et al., 2015), The student agent can then sample from this collection of previously encountered environments for training. Additionally, Parker-Holder et al. (2022) adopt a different strategy by using predetermined starting points for environments and gradually increasing complexity. They manually divide the environment design space into different difficulty levels and employ human-defined edits to generate similar environments with high learning potentials. Their algorithm, ACCEL, is currently the state-of-the-art (SOTA) in the field, and we use it as a baseline in our experiments.

### 2.2 Diffusion Probabilistic Models

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) are a specific type of generative model that learns the data distribution from a dataset. Recent advances in diffusion-based models, including Langevin dynamics and score-based generative models, have shown promising results in various applications, such as time series forcasting (Tashiro et al., 2021), robust learning (Nie et al., 2022), anomaly detection (Wyatt et al., 2022) as well as synthesizing high-quality images from text desciptions (Nichol et al., 2021; Saharia et al., 2022). These models can be trained using standard optimization techniques, such as stochastic gradient descent, making them highly scalable and easy to implement.

In a diffusion probabilistic model, we assume a $D$-dimensional random variable $\boldsymbol{x} \in \mathbb{R}^D$ with an unknown distribution $q_0(x_0)$. Diffusion Probabilistic model involves two Markov chains: a forward chain $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ that perturbs data to noise, and a reverse chain that converts noise back to data. The forward chain is typically designed to transform any data distribution into a simple prior distribution (e.g., standard Gaussian) by considering perturb data with Gaussian noise of zero mean and $\beta_t$ variance for $T$ steps:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}) \quad q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \Pi_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

where $t \in \{0, \ldots, T\}$ and $0 < \beta_{1:T} < 1$ denote the noise scale scheduling. At the end, $\boldsymbol{x}_T \to \mathcal{N}(0, \boldsymbol{I})$ will converge to isometric Gaussian noise. According to the rule of the sum of normally distributed random variable, the choice of Gaussian provides a closed-form solution to generate arbitrary time-step $\boldsymbol{x}_t$ through:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. The reverse chain $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ reverses the forward process by learning transition kernels parameterized by deep neural networks. New data points are subsequently generated by first sampling a random vector from the prior distribution, followed by ancestral sampling through the reverse Markov chain. Specifically, considering the Markov chain parameterized by $\boldsymbol{\theta}$, denoising arbitrary Gaussian noise into clean data samples can be written as:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t))$$

It uses the Gaussian form $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ because the reverse process has the identical function form as the forward process when $\beta_t$ is small (Sohl-Dickstein et al., 2015).

Ho et al. (2020) show that the training to maximize the log-likelihood $\int q(x_0) \log p_{theta}(x_0 dx_0)$ is equivalent to minimize a re-weighted evidence lower bound (ELBO) that fits the noise. They derive

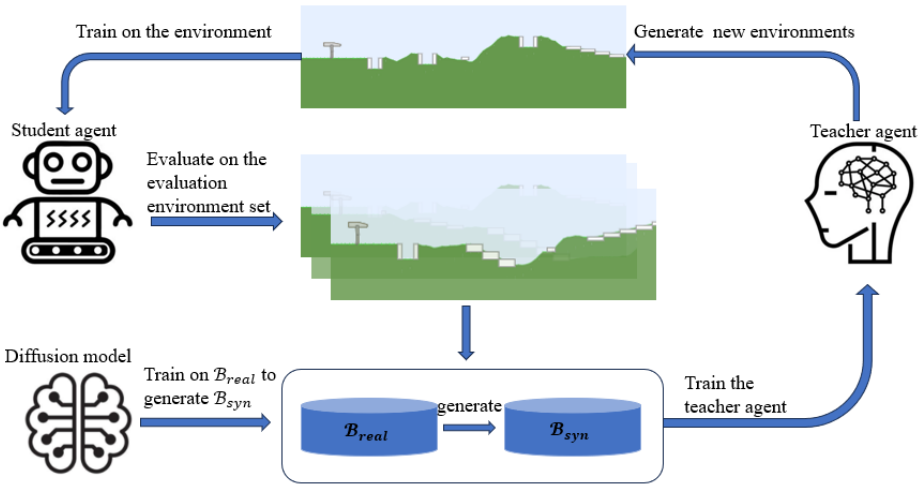


Figure 1: The overall framework of *SHED*.

the final objective by parameterization and simplication:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}_0,t,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]$$

where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is a function approximator to predict $\boldsymbol{\epsilon}$ from $\boldsymbol{x}_t$.

## 3 Approach

In this section, we formally present *S*ynthesis-enhanced *H*ierarchical *E*nvironment *D*esign (*SHED*) as a novel framework in UED and discuss it in detail. We begin by describing the procedure for generating a distribution of environments through the hierarchical framework and then show how the generative model may be adapted to our continual training process.

### 3.1 Hierarchical Environment Design

Similar to previous work ((Dennis et al., 2020)), our objective is to generate a sequence of environments that effectively support the continual learning of the student agent. Presently, mainstream approaches to the environment generation process (PLR (Jiang et al., 2021a),ACCEL (Parker-Holder et al., 2022)) often rely on domain randomization, which involves generating random environments. However, these approaches face challenges in effectively exploring the action space, particularly when dealing with a large parameter space, and they are not able to leverage previously discovered environmental structures. Motivated by the principles of the PAIRED (Dennis et al., 2020) algorithm, we adopt an RL-based approach for the environment generation process.

At the core of *SHED* is the hierarchical MDP framework, consisting of an upper-level RL teacher policy and a lower-level student policy. Specifically, our framework involves specifying the upper-level teacher policy, $\Lambda : \boldsymbol{\Pi} \rightarrow \Delta(\theta)$, where $\boldsymbol{\Pi}$ represents the set of possible student policies, and $\theta$ signifies the range of potential environmental parameters. Diverging from RL teacher in the PAIRED algorithm, which takes inputs comprising the current fully observed state of the environment, the current time step $t$, and a random vector $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$, our method first approximates the student agent's policy by assessing its performance across a diverse set of environments. This input provides a more accurate reflection of the student's current capability level, enabling the teacher to consistently design environments that effectively support continuous training. We now elaborate on the specifics of our hierarchical framework.

Consider an environment generation system governed by discrete-time dynamics. Given the current batch of generated environments, each environment is a fully specified environment for students, characterized by a Partially Observable Markov Decision Process (POMDP), which is defined by a tuple $< \mathbb{A}, \mathbb{O}, \mathbb{S}^{\boldsymbol{\theta}}, \mathcal{P}^{\boldsymbol{\theta}}, \mathcal{I}^{\boldsymbol{\theta}}, \mathcal{R}^{\boldsymbol{\theta}}, \gamma >$, where $\mathbb{A}$ represents the set of actions, $\mathbb{O}$ is the set of observations, $\mathbb{S}^{\boldsymbol{\theta}}$ is the set of states determined by the environment parameters $\boldsymbol{\theta}$, similarly, $\mathcal{P}^{\boldsymbol{\theta}}$ is the

environment-dependent transition function, and $\mathcal{I}^{\boldsymbol{\theta}} : \mathbb{A} \rightarrow \mathbb{O}$ is the environment-dependent observation function, $\mathcal{R}^{\boldsymbol{\theta}}$ is the reward function, and $\gamma$ is the discount factor. The students is trained to maximize their cumulative reward $V^{\boldsymbol{\theta}}(\pi) = \sum_{t=0}^{T} \gamma^t r_t$ for the current environment under a finite time horizon $T$. This forms the foundation of our lower-level student MDP.

Once training within the current batch of environments is completed, we evaluate the student's performance across a diverse set of evaluation environments. The vector of performance metrics serves as an approximation of the student's policy embedding. Therefore, our teacher policy, $\Lambda$, maps the current student's policy to the most suitable environment for enhancing its performance, i.e., $\Lambda : \pi \rightarrow \Delta(\theta)$, where $\pi \in \Pi$ represents the current student policy. From the teacher's perspective, the evolving of student's policy can be represented by the upper-level MDP, characterized as tuple $< \mathbb{S}, \mathbb{A}, P, R, \gamma >$. Here, $\mathbb{S}$ represents the student policy space, which is approximated by its performance across the evaluation environments set. and $\mathbb{A}$ is the teacher's action space, which corresponds to the environment parameter space. $P$ is the transition function of student policy's performances, and $R$ is the reward function. This hierarchical approach enables us to systematically measure and enhance the performance of the student agent across various environments and adapt the training process accordingly.

However, it's worth noting that collecting the teacher's experience in this hierarchical framework is notably resource-intensive and time-consuming, since each transition within the upper-level teacher MDP encompasses a/several full lower-level student MDPs. In the following section, we will formally introduce a generative model designed to streamline the collection of upper-level MDP experience. This will enable us to train our teacher policy more efficiently and effectively. The overall framework is shown in Figure 1, and the pseudo-code is provided in Algorithm 1.

**Algorithm 1** *SHED*

**Input:** real data ratio $r \in [0, 1]$, evaluate environment set $\boldsymbol{\theta}^{eval}$, reward function $R$ for teacher;

**Initialize:** real replay buffer $\boldsymbol{\mathcal{B}}_{real} = \emptyset$, synthetic replay buffer $\boldsymbol{\mathcal{B}}_{syn} = \emptyset$, diffusion model $M$, teacher policy $\Lambda$;
**for** episode $ep = 1, \ldots, K$ **do**
  Initialize student policy $\pi$
  Evaluate $\pi$ on $\boldsymbol{\theta}^{eval}$ and get initial state $s$
  **for** step $t = 0, \ldots, T$ **do**
    use $\Lambda$ to generate new batch of environment parameters $\boldsymbol{\theta}$, and create $\mathcal{M}_{\boldsymbol{\theta}}(\pi)$
    train the $\pi$ on $\mathcal{M}_{\theta}$ to maximize $V^{\boldsymbol{\theta}}$
    evaluate $\pi$ on $\boldsymbol{\theta}^{eval}$ and get next state $s'$
    compute teacher's reward $r_t$ according to $R$
    add teacher's experience $(\boldsymbol{s}, \boldsymbol{\theta}, r_t, \boldsymbol{s'})$ to $\boldsymbol{\mathcal{B}}_{real}$
    train $M$ with samples from $\boldsymbol{\mathcal{B}}_{real}$
    generate samples from $M$ and add them to $\boldsymbol{\mathcal{B}}_{syn}$
    train $\Lambda$ on samples from $\boldsymbol{\mathcal{B}}_{real} \bigcup \boldsymbol{\mathcal{B}}_{syn}$ mixed with ratio $r$
    set $\boldsymbol{s} = \boldsymbol{s'}$;
  **end for**
**end for**

**Output:** The teacher policy $\Lambda$, student policy $\pi$, and diffusion model $M$

### 3.2 Generative Trajectory Modeling

Here, we describe how to take advantage of recent advancements in generative models, such as the diffusion model to generate synthetic trajectories that can be used to help train the teacher agent, resulting in reducing the resource-intensive and time-consuming collection of upper-level teacher's experiences.

In this work, we deal with two distinct types of timesteps: one for the diffusion process and another for reinforcement learning. We use subscripts $i \in 1, \ldots, N$ to represent diffusion timesteps and subscripts $t \in 1, \ldots, T$ to represent trajectory timesteps.

Our diffusion model, denoted as $M$, may be utilized to generate synthetic teacher experiences by continually training the diffusion model on newly collected experiences. Let the tuple $(\boldsymbol{s}, \boldsymbol{a}, r_t, \boldsymbol{s'})$

denote synthetic experience generated by the diffusion model. We begin by randomly sampling the observed state from the real experience buffer $\boldsymbol{s} \sim \boldsymbol{\mathcal{B}}_{real}$, and the generation of $(\boldsymbol{s}, \boldsymbol{\theta}, r_t, \boldsymbol{s}')$ can be divided into three steps: 1) generate actions $\boldsymbol{a}$ (which corresponds to the environment parameters $\boldsymbol{\theta}$); 2) given the state-action pair $\boldsymbol{\tau} = (\boldsymbol{s}, \boldsymbol{a})$, generate the next state $\boldsymbol{s}'$; 3) compute the reward $r_t$ based $(\boldsymbol{s}, \boldsymbol{\theta}, \boldsymbol{s}')$.

In the first step, we can either generate our action through direct sampling from

$$\boldsymbol{a} \sim \Lambda(\boldsymbol{a}|\boldsymbol{s})$$

or represent the action generation via the reverse process of a conditional diffusion model as

$$\boldsymbol{a} \sim p_{\boldsymbol{\theta}}(\boldsymbol{a}_{0:N}|\boldsymbol{s})$$

where $\boldsymbol{a}_0$ at the end sample of the reverse chain is the action that we want. This process is similar to the way we generate $\boldsymbol{s}'$; therefore, we omit it here. The first one is straightforward to implement because we can stochastically generate different actions $\boldsymbol{a}$ directly based on the existing teacher's policy $\Lambda$.

For the second step, we generate our next state $\boldsymbol{s}$ via the reverse process of a conditional diffusion model as:

$$\boldsymbol{s}' \sim p_{\boldsymbol{\theta}}(\boldsymbol{s}'_{0:N}|\boldsymbol{\tau}) = \mathcal{N}(\boldsymbol{s}'_N; \mathbf{0}, \boldsymbol{I}) \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{s}'_{i-1}|\boldsymbol{s}_i, \boldsymbol{\tau})$$

and $\boldsymbol{s}'_0$ at the end of sampling in the reverse chain is the generated synthetic next state. As shown in Section 2.2, $p_{\boldsymbol{\theta}}(\boldsymbol{s}'_{i-1}|\boldsymbol{s}_i, \boldsymbol{\tau})$ could be modeled as a Gaussian distribution $\mathcal{N}(\boldsymbol{s}_{i-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}'_i, \tau, I), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \tau, i))$. Similar to Ho et al. (2020), we parameterize $p_{\boldsymbol{\theta}}(\boldsymbol{s}'_{i-1}|\boldsymbol{s}_i, \boldsymbol{\tau})$ as a noise prediction model with the covariance matrix fixed as

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \tau, i) = \beta_i \boldsymbol{I}$$

and mean is

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}'_i, \tau, i) = \frac{1}{\sqrt{\alpha_i}} \left( \boldsymbol{s}'_i - \frac{\beta_i}{\sqrt{1-\bar{\alpha}_i}} \boldsymbol{\epsilon}(\boldsymbol{s}'_i, \boldsymbol{\tau}, \boldsymbol{i}) \right)$$

Following Wang et al. (2022), we begin by sampling $\boldsymbol{s}_N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and then proceed with the reverse diffusion chain $p_{\boldsymbol{\theta}}(\boldsymbol{s}'_{i-1}|\boldsymbol{s}'_i)$ for $i = N, \ldots, 1$, which is parameterized by $\boldsymbol{\theta}$ as follows:

$$\frac{\boldsymbol{s}'_i}{\sqrt{\alpha_i}} - \frac{\beta_i}{\sqrt{\alpha_i(1-\bar{\alpha}_i)}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{s}'_i, \boldsymbol{\tau}, \boldsymbol{i}) + \sqrt{\beta_i}\boldsymbol{\epsilon}, \tag{1}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. When $i = 1$, $\boldsymbol{\epsilon}$ is set as $\mathbf{0}$ to improve the sampling quality. We employ a similar simplified objective, as proposed by Ho et al. (2020) to train our conditional $\boldsymbol{\epsilon}$ through the following process:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{\tau}, \boldsymbol{s}') \sim \boldsymbol{\mathcal{B}}_{real}, i \sim \mathcal{U}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[ \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\sqrt{\bar{\alpha}_i}\boldsymbol{s}' + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \boldsymbol{\tau}, i)\|^2 \right] \tag{2}$$

Here $\mathcal{U}$ represents a uniform distribution over the discrete set $\{1, \ldots, N\}$. The loss function of the diffusion model, denoted as $\mathcal{L}_{simple}(\boldsymbol{\theta})$ is essentially a behavior-cloning loss. it aims to generate the synthetic teacher's experience $(\boldsymbol{s}, \boldsymbol{\theta}, r_t, \boldsymbol{s}')$. Furthermore, this method enables us to learn from the human-generated experience as it is sampling-based and only requires taking random samples from both $\boldsymbol{\mathcal{B}}_{real}$ and the current policy without requiring to know the behavior policy.

The Gaussian assumption holds true primarily under the condition of the infinitesimally limit of small denoising steps (Sohl-Dickstein et al., 2015). Consequently, this necessitates a substantial number of steps in the reverse process. Furthermore, recent research byXiao et al. (2021) have demonstrated that enabling denoising with large steps can reduce the total number of denoising steps, $N$. Consequently, in order to expedite the relatively slow reverse sampling process outlined in Equation 3.2 (as it requires computing $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ networks $N$ times), we use a small value of $N$, while simultaneously setting $\beta_{\min} = 0.1$ and $\beta_{\max} = 10.0$. Similar to Wang et al. (2022), we define:

$$\beta_i = 1 - \alpha_i = 1 - \exp\left( \beta_{\min} \times \frac{1}{N} - 0.5(\beta_{\max} - \beta_{\min}) \frac{2i-1}{N^2} \right)$$

This noise schedule is derived from the variance-preserving Stochastic Differential Equation proposed by Song et al. (2020).

In the final step, after obtaining the generated action $\boldsymbol{a}$, and next state $\boldsymbol{s}'$, we compute the reward $r_t$ using the teacher's reward function $R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$. The specifics of how the reward function is chosen will be explained in the following section.

### 3.3 Rewards and Choice of evaluate environments

Our upper-level teacher policy generates environments tailored specifically for the lower-level student policy, aligning with the most suitable environments to improve the general capability of the lower-level student policy. That is, at each time step $t$, the upper-level teacher policy generates a batch of environments, indicating its desire for the lower-level student agent to undergo training within these generated environments to enhance the student's overall capability across a diverse set of evaluation environments.

The selection of suitably diverse evaluation environments is crucial because they reflect the agent's general capabilities and serve as an approximation of the policy's embedding. Fontaine & Nikolaidis (2021) propose the use of quality diversity (QD) optimization to generate collections of high-quality environments that exhibit diversity for the resulting agent behaviors. Similarly, Bhatt et al. (2022) introduce a QD-based algorithm for dynamically designing such evaluation environments based on the current agent's behavior. However, it's worth noting that this QD-based approach can be tedious and time-consuming, and it heavily relies on the given agent policy.

Given these considerations, it is natural to take advantage of the domain randomization algorithm, as it has demonstrated compelling results in generating diverse environments and training generally capable agents. In our approach, we initially discretize environment parameters into different ranges, then randomly sample from these ranges, and combine these parameters to generate evaluation environments. This method has yielded promising empirical results. We define the reward for the upper-level teacher policy as a parameterized function based on the improvement in student performance on the evaluation environment after training in the current generated environment:

$$R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = \|\boldsymbol{s} - \boldsymbol{s}'\|_1$$

This rewards upper-level teachers for taking action to create the right environment to improve the overall performance of students across diverse environments. However, such a reward function primarily focuses on the overall performance of the student agent in the assessment environment. The teacher agent can gain higher rewards by sacrificing student performance in one subset of evaluation environments to improve student performance in another subset, which conflicts with our objective. Therefore, we need to consider fairness in the reward function to avoid this situation and ensure that the generated environment can improve student's general capabilities. Similar to Elmalaki (2021), we build our fairness metric on top of the change in student's performance in each evaluation environment, denoted as $u_i = s_i' - s_i$, and we have $\bar{u} = \frac{1}{n}\sum_{i=1}^{n} u_i$. We then measure the fairness of the teacher's action using the coefficient of variation of student performances:

$$cv(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = \sqrt{\frac{1}{n-1}\sum_{i}\frac{(u_i - \bar{u})^2}{\bar{u}^2}} \tag{3}$$

A teacher is considered to be fair if and only if the $cv$ is smaller. As a result, our reward function is:

$$R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = \|\boldsymbol{s} - \boldsymbol{s}'\|_1 - c \cdot cv(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') \tag{4}$$

Here $c$ is the coefficient that balances the weight of fairness in the reward function (We set a small value to this, such as 0.05). This reward function motivates the teacher agent to generate training environments that can improve student's general capabilities.

## 4 Experiments

In our experiments, we first use a mathematical example to assess the quality of the synthetic experiences generated by the diffusion model. Subsequently, we compare *SHED* to the leading prior

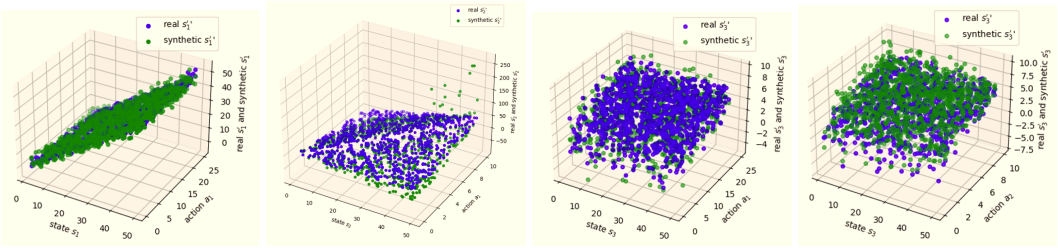

Figure 2: The distribution of the real $\boldsymbol{s}'$ and the synthetic $\boldsymbol{s}'$ conditioned on $(\boldsymbol{s}, \boldsymbol{a})$.

approaches in UED. Our experiments are conducted on two domains: Lunar Lander and a modified BipedalWalker environment. Our primary comparisons involve *SHED* against five baselines: domain randomization (Tobin et al., 2017), ACCEL (Parker-Holder et al., 2022) (with slight modifications that it does not revisit the previously generated environments), PAIRED (Dennis et al., 2020), and h-MDP (our proposed hierarchical approach without diffusion model aiding in training). In all cases, we train a student agent via Proximal Policy Optimization (PPO (Schulman et al., 2017), and train the teacher agent via Deterministic policy gradient algorithms(DDPG (Silver et al., 2014)), because it is more sample efficient and can learn from both real experience buffer or the synthetic experience buffer.

### 4.1 Analyzing the accuracy of Diffusion model

We begin by investigating *SHED*'s ability to assist in collecting experiences for the upper-level MDP teacher. This involves the necessity for *SHED* to prove its ability to accurately generate synthetic experiences for teacher agents. To check the quality of these generated synthetic experience, we employ a diffusion model to simulate some data for validation (even though Diffusion models have demonstrated remarkable success across vision and NLP tasks).

We design the following experiment: given input $\boldsymbol{s} = [s_1, s_2, s_3]$, which represents the student's current performances on the evaluation environment set, and another input $\boldsymbol{a} = [a_1, a_2]$, which represents action of the teacher(environment parameters), we use the function $\boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a})$ to simulate the student training process, where $\boldsymbol{s}' = [s_1', s_2', s_3']$ is the student's updated performances after training in the environment (generated by $\boldsymbol{a}$). The specific dynamic relationship is as follows:

$$s_1' = \begin{cases} s_1 + 2|s_1 - 2a_1| + \epsilon & \text{if } 1 < |s_1 - 2a_1| < 2 \\ s_1 - 1 + \epsilon & \text{otherwise} \end{cases} \quad s_2' = \begin{cases} s_2 + \exp(s_2 - 5a_2) + \epsilon & \text{if } |s_2 - 5a_2| < 2 \\ s_2 - |s_2 - 5a_2| + \epsilon & \text{otherwise} \end{cases}$$

$$\text{and} \quad s_3' = \begin{cases} s_3 + \log(a_1 * a_2 - s_3) + \epsilon & \text{if } 1 < \log(a_1 * a_2 - s_3) < 5 \\ s_3 - |\log(a_1 * a_2 - s_3)| + \epsilon & \text{otherwise} \end{cases}$$

We first train our diffusion model on the real $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ dataset generated by function $f(\boldsymbol{s}, \boldsymbol{a})$. We then randomly sample 1000 combination data of $\boldsymbol{s}$ and $\boldsymbol{a}$, input them into $f(\boldsymbol{s}, \boldsymbol{a})$ to get the real $\boldsymbol{s}'$. The trained diffusion model is then used to generate the synthetic $\boldsymbol{s}'$ conditioned on $(\boldsymbol{s}, \boldsymbol{a})$ pair.

The experiment results are presented in Figure 2. The results show that the generative model can effectively capture the distribution of real experience, and the generated synthetic experience is very close to the real dynamic relationship $\boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a})$, thereby validating that the diffusion model can generate useful experience conditioned on $(\boldsymbol{s}, \boldsymbol{a})$. It is important to note that the marginal distribution derived from the reverse diffusion chain provides an implicit, expressive distribution, such distribution has the capability to capture complex distribution properties, including skewness and multi-modality.

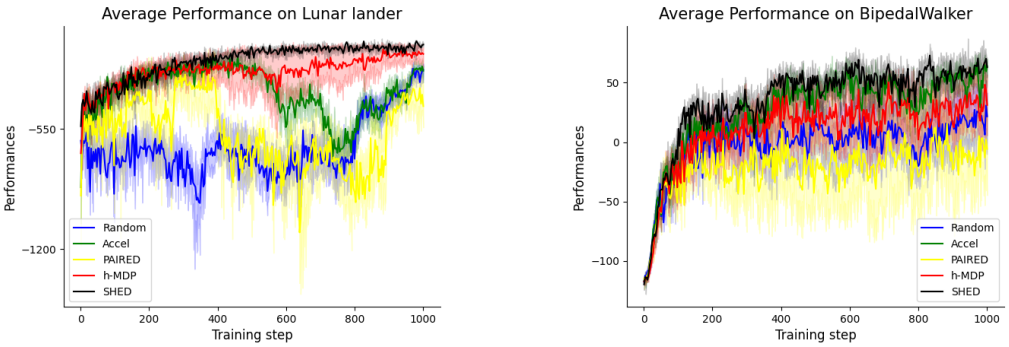


Figure 3: (*left*) Zero-shot transfer performance on the test environments in the Lunar lander environment. (*right*) Zero-shot transfer performance on the test environments in the BipedalWalker.

### 4.2 LUNARLANDER

We next evaluate our approach in the Lunar Lander environment, a classic rocket trajectory optimization problem. In this domain, student agents are tasked with controlling a lander's engine to safely land the vehicle. Before the start of each episode, teacher algorithms determine the environment parameters that are used to generate environments in a given play-through, which includes gravity, wind power, and turbulence power. These parameters directly alter the difficulty of landing the vehicle safely. The state is an 8-dimensional vector, which includes the coordinates of the lander, its linear velocities, its angle, its angular velocity, and two booleans that represent whether each leg is in contact with the ground or not.

We train the student agent for 1e6 environment time steps and periodically evaluate the agent in test environments that are not present in the training or evaluation environments (performances vector on the evaluation environments are used as the teacher agent's state). The parameters for the test environments are randomly generated and fixed during training. We report the experiment result on the left side of Figure 3. As we can see, student agents trained under *SHED* consistently outperform other baselines and have the minimal variance in transfer performance. ACCEL experiences a performance dip in the middle during training. This phenomenon could potentially be attributed to the inherent challenge of manually design the appropriate level of difficulty in the parameter space.

In the appendix, we detail how the performance of different methods changes in each testing environment during training. Furthermore, we conduct experiments to show how the algorithm performs under different settings, such as a longer training time horizon, or a larger weight of cv fairness rewards. We noticed an interesting finding: when fairness reward has a high weightage, our algorithm tends to generate environments at the onset that lead to a rapid decline and subsequent improvement in students' performance across all test environments. This is done to avoid acquiring a substantial negative fairness reward and thereby maximize the teacher's cumulative reward. Notably, the student's final performance still surpasses other baselines. See the appendix for detailed results.

### 4.3 BIPEDALWALKER

Finally, we evaluate *SHED* in the modified BipedalWalker from Parker-Holder et al. (2022). In this domain, the student agent is required to control a bipedal vehicle and navigate across a terrain, and the student agent receives a 24-dimensional proprioceptive state with respect to its lidar sensors, angles, and contacts. The teacher is tasked to select eight variables (including ground roughness, the number of stair steps, min/max range of pit gap width, min/max range of stump height, and min/max range of stair height) in order to generate the corresponding terrain.

We use the simialr experiment settings in prior UED works, we train all the algorithms for 1e7 environment time steps, and then evaluate their generalization ability on ten distinct test environments in Bipedal-Walker domain. The parameters for the test environments are randomly generated and fixed during training. As shown in Figure 3, our proposed method *SHED* surpasses all other baselines, and

achieves performance levels nearly on par with the SOTA (ACCEL). Meanwhile, PAIRED suffers from a considerable degree of variance in its performance.

## 5 CONCLUSION

In this paper, we have introduced an adaptive approach for efficient training of a generally capable agent within a finite time horizon. Our approach is general, utilizing an upper-level MDP teacher agent that can guide the training of the lower-level MDP student agent. Our hierarchical framework can incorporate any techniques developed from existing UED works, such as prioritized level replay (revisiting environments with high learning potential). Furthermore, we have described a method to assist the experience collection for the teacher when it is trained in an off-policy manner. Our experiment demonstrates that our method outperforms prior UED methods, underscoring its effectiveness as a curriculum-based learning approach within the UED framework.

## REFERENCES

Varun Bhatt, Bryon Tjanaka, Matthew Fontaine, and Stefanos Nikolaidis. Deep surrogate assisted generation of environments. *Advances in Neural Information Processing Systems*, 35:37762–37777, 2022.

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

Salma Elmalaki. Fair-iot: Fairness-aware human-in-the-loop reinforcement learning for harnessing human variability in personalized iot. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, pp. 119–132, 2021.

Matthew Fontaine and Stefanos Nikolaidis. Differentiable quality diversity. *Advances in Neural Information Processing Systems*, 34:10040–10052, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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

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

Wenjun Li, Pradeep Varakantham, and Dexun Li. Effective diversity in unsupervised environment design. *arXiv preprint arXiv:2301.08025*, 2023.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. *arXiv preprint arXiv:2205.07460*, 2022.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. *arXiv preprint arXiv:2203.01302*, 2022.

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. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

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

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

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. Pmlr, 2014.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

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 *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Julian Wyatt, Adam Leach, Sebastian M Schmon, and Chris G Willcocks. Anoddpm: Anomaly detection with denoising diffusion probabilistic models using simplex noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 650–656, 2022.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

Matthieu Zimmer, Claire Glanois, Umer Siddique, and Paul Weng. Learning fair policies in decentralized cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 12967–12978. PMLR, 2021.

# A Appendix

You may include other additional sections here.

## A.1 Generative trajectory modeling

The trajectory modeling can be represented via the reverse process of a conditional diffusion model as

$$\boldsymbol{a} \sim \Lambda(\boldsymbol{a}|\boldsymbol{s}) = p_{\boldsymbol{\theta}}(\boldsymbol{a}^{0:N}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}^N; \mathbf{0}, \boldsymbol{I}) \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{a}^{i-1}|\boldsymbol{a}^i, \boldsymbol{s})$$

where $\boldsymbol{a}^0$ at the end sample of the reverse chain is the action that we want.

### A.2 FAIRNESS

The current main focus is on the OOD performance with respect to the average per-student efficiency measure (e.g., worst performance of each student). Thus, for teacher-student framework, fairness is a key factor to ocnsider in their design for their successful deployments and operations.

Fairness is a multifaceted concept, which can refer to or include different aspects, i.e., Pareto-efficiency, "envy-freeness", or proportionality among students. Given the importance of this notion, it has been investigated in various applications.

For example:

- Zimmer et al. (2021) define fairness that refers to the combination of three aspects: impartiality, equity, and efficiency. The author consider the complex control problems, but in the cooperative setting. Impartiality corresponds to the "equal treatment of equals" principle, which is arguably one of the most important pillars of fairness. They assume that all users are identical and should therefore be treated similarly. Equity is based on the Pigou-Dalton principle, which states that a reward transfer from a better-off user to a worse-off user yields a fairer solution. Efficiency states that between two feasible solutions, if one solution is (weakly or strictly) preferred by all users, then it should be preferred to the other one.
- Elmalaki (2021) propose fairness in multi-human IoT applications.

Similar to Elmalaki (2021), we can build our fairness on top of the utility. we define the utility of each student agent as :

$$u_t = \frac{1}{T}\sum_{1}^{T}\gamma^t r_t \tag{5}$$

We then can measure the fairness of the RL generator using the coefficient of variation of student utilities:

$$cv = \sqrt{\frac{1}{n-1}\sum_{i}\frac{(u_i - \bar{u})^2}{\bar{u}^2}} \tag{6}$$

A teacher is said to be fair if and only if the $cv$ is smaller. Note that we can also use the marginal benefit of each agent as the utility (based on our previous work) Fairness is a multifaceted concept, which can refer to or include different aspects.

### A.3 ADDITIONAL EXPERIMENTS

We further conduct experiments in Lunar lander under a longer time horizon. The results are provided in Figure 4.

we also conduct experiments to show how the algorithm performs under different settings, such as a larger weight of cv fairness rewards ($c = 1$). The results are provided in Figure 5. We noticed an interesting finding: when fairness reward has a high weightage, our algorithm tends to generate environments at the onset that lead to a rapid decline and subsequent improvement in students' performance across all test environments. This is done to avoid acquiring a substantial negative fairness reward and thereby maximize the teacher's cumulative reward. Notably, the student's final performance still surpasses other baselines.

We further show in detail how the performance of different methods changes in each testing environment during training (see Figure 6 and Figure 7 ).

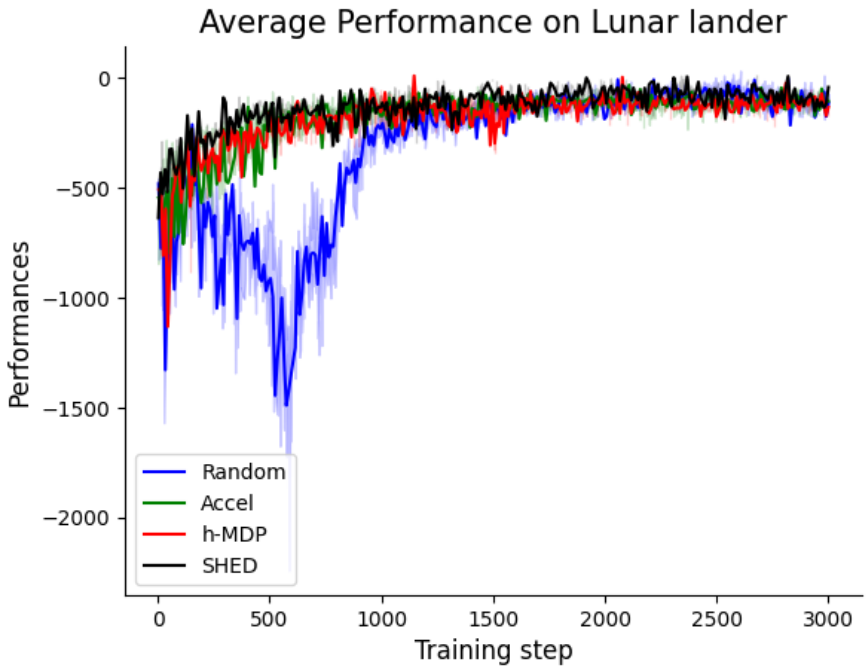


Figure 4: Zero-shot transfer performance on the test environments under a longer time horizon in Lunar lander environments.

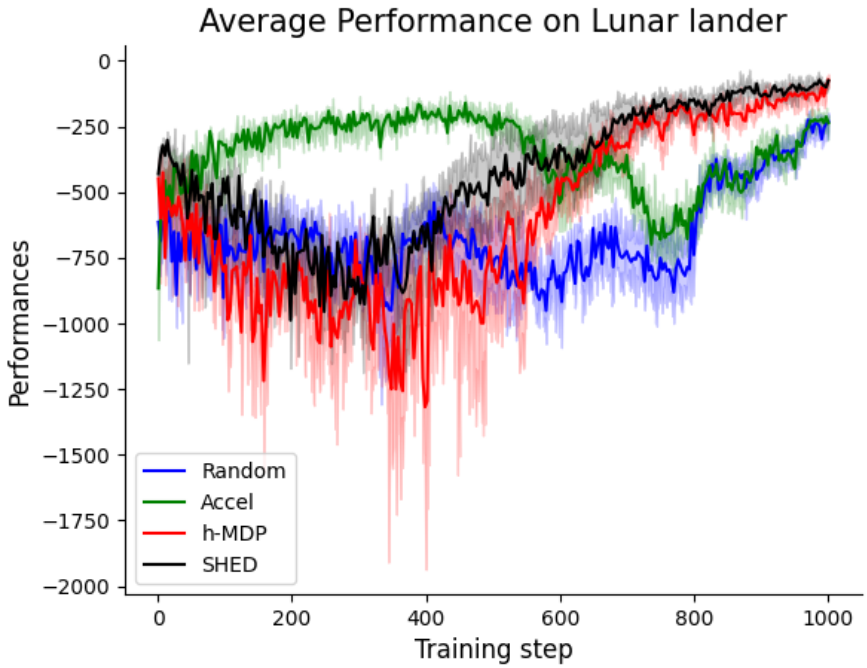


Figure 5: Zero-shot transfer performance on the test environments with a larger $cv$ value coefficient in Lunar lander environments.

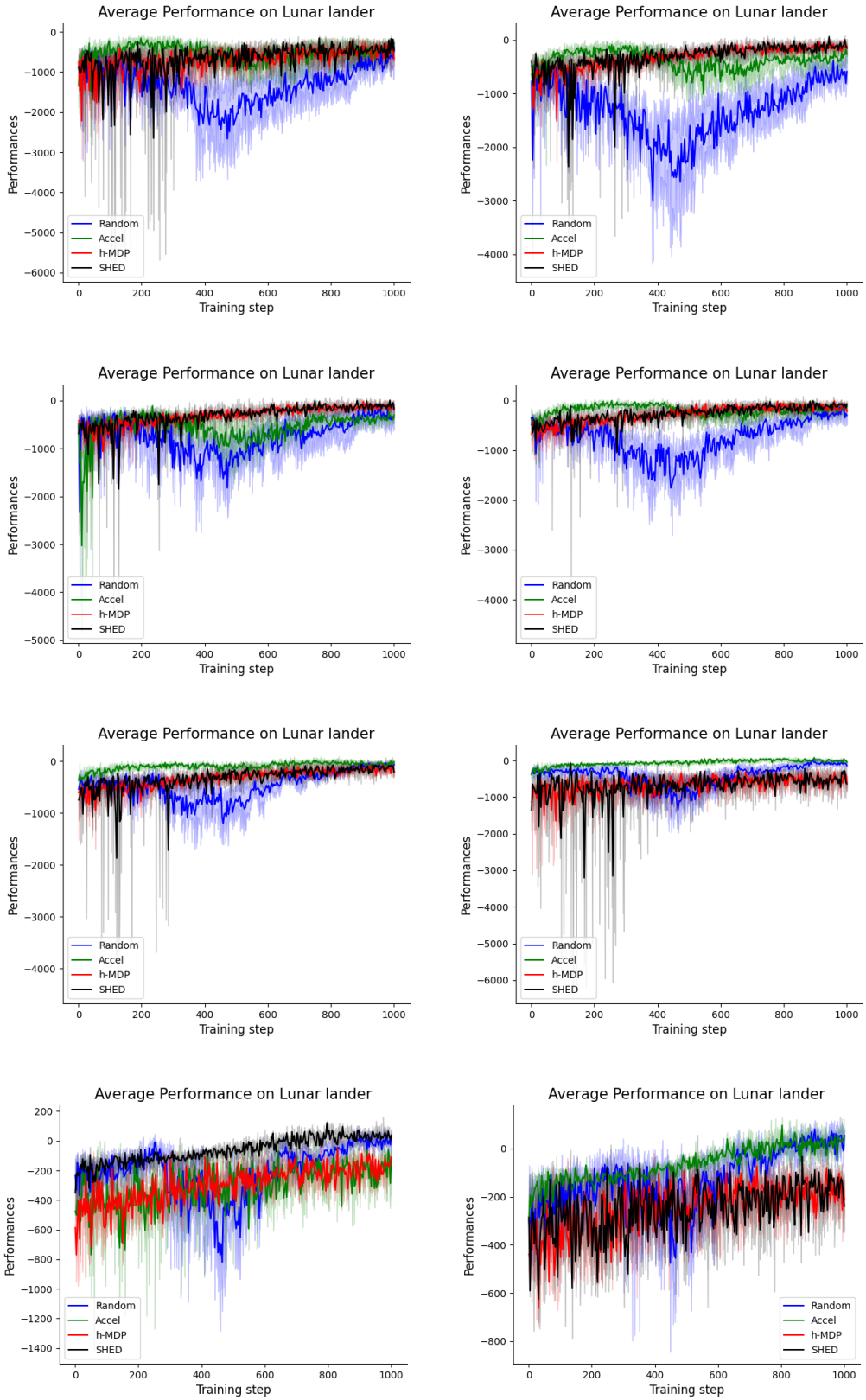


Figure 6: Caption for all images

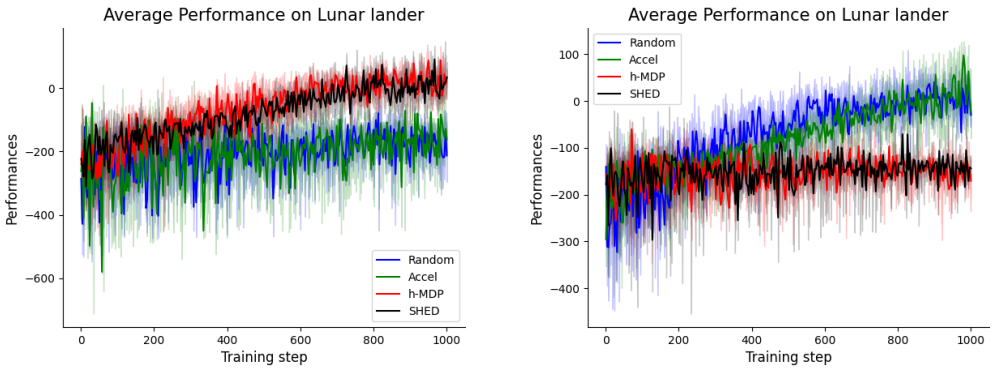


Figure 7: Caption for all images