CLUTR: CURRICULUM LEARNING VIA UNSUPERVISED TASK REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning (RL) algorithms are notorious for sample inefficiency and lack of generalization. Recently, a class of unsupervised curriculum learning algorithms, called Unsupervised Environment Design(UED), have shown promising generalization to unseen tasks by training agents on a diverse distribution of synthetic tasks. Contemporary UED algorithms employ a teacher agent which generate tasks from scratch by generating sequences of environment parameters (e.g., location of obstacles in a navigation task) and is trained with a minimax regret objective. However, these UED algorithms are sample inefficient. First, training the teachers are hard due to the long-horizon credit assignment in an exponential parameter space. More importantly, they simultaneously learn an implicit task manifold as well as how to navigate it to create an effective curriculum, which is difficult due to the task manifold evolving over time. We introduce CLUTR, an UED method augmenting PAIRED — one of the most popular and principled UED method — with unsupervised task-representation learning. CLUTR teacher induces a curriculum by sampling concrete tasks from a pretrained generative latent task space which decouples learning task representation from curriculum learning objective and can also be formulated with a single step RL avoiding the long-horizon credit assignment. The pretrained task-space is trained from randomly generated parameters without any interaction with the simulator and also solves the parameter space explosion problem faced by other parameter-space teachers. Our experimental results show CLUTR outperforms previous SOTA UED methods both, in terms of generalization and sample-efficiency, in MiniGrid and CarRacing environments.

1 INTRODUCTION

Reinforcement Learning have shown exciting progress in the past decade solving Atari Mnih et al. (2015), Dota Berner et al. (2019), Go Silver et al. (2016) and other challenging domains. However, RL is also notorious for sample inefficiency and lack of generalization. Curriculum Learning(CL) algorithms have shown to improve sample efficiency by training the agents on a sequence of tasks[1], starting from the easy ones and then progressively increasing the difficulty. CL algorithms always try to present "just-the-right-amount-of-challenge" to the agents by generating tasks at the frontier of the agent's capabilities. A more recent class of Unsupervised CL algorithms, called Unsupervised Environment Design (UED), have shown impressive zero-shot generalization by training the agents on a diverse distribution of synthetic tasks generated by sampling from a given underspecified paramater space of the environment. PAIRED(Dennis et al. (2020)), which introduced UED, is one of the most popular and principled UED algorithms. PAIRED trains a teacher agent using adversarial training, where the agent to be trained (also called protagonist) and a second agent, called the Antagonist, are employed and trained in same tasks generated by the teacher agent, while the teacher itself optimizes the regret defined as the difference of the antagonist and the agent's return. Theoretically this regret optimization incentivizes the PAIRED teacher to generate feasible but challenging tasks for the agent and is adopted by the contemporary UED algorithms (e.g., Parker-Holder et al. (2022), Jiang et al. (2021a)).

[1]To avoid confusion, throughout this paper we will use the term Environment as the Simulator and Task to refer a concrete MDP from an underspecified one—often referred as Environment in UED literature—e.g., location of obstacles for navigation

In spite of the generalization capabilities of PAIRED, in practice it is sample inefficient. Attempts have been made to improve it by augmenting it with other techniques such as level replay(Jiang et al. (2021a)). However, training a regret-based teacher is hard, often deemed as the Achilles' heel of UED algorithms(Parker-Holder et al. (2022)), which involves a long-horizon credit assignment in an exponential parameter space. The contemporary UED algorithm teachers are RL agents, which typically generate a fixed-length sequence of integers (e.g., location of the obstacles for navigation) and at the end receives the regret as a reward and hence need to solve a long horizon sparse-reward credit assignment problem. Additionally, the teacher agents also face a combinatorial explosion as they often represent sets as sequences—e.g., for a navigation task the same set of obstacles can be generated by a plethora of different sequence of parameters, in order of the factorial of the number of blocks. More importantly, the contemporary teachers simultaneously learn a task manifold as well as navigating it to create an effective curriculum, which is difficult due to the task manifold evolving non-stationarily over time. The task manifold is an important aspect of curriculum learning algorithms enabling the teachers to find tasks at the frontier of agents' capabilities. Moreover, this task manifold is learned indirectly while optimizing for regret without any explicit regards to the actual task structures.

To address the above mentioned challenges we present CLUTR: Curriculum Learning via Unsupervised Task Representation Learning. At the heart of our method is a generative model that learns a latent representation of tasks in an unsupervised manner. The CLUTR teacher samples from the latent task space and the generative model is used to obtain the concrete parameters of the task. CLUTR learns a curriculum by navigating the stationary task-manifold by optimizing the curriculum learning objective (i.e. maximizing regret) similar to PAIRED. In contrast to the PAIRED teacher, which tries to build the tasks from scratch one parameter at a time, CLUTR teacher can be seen as 'choosing' concrete tasks from the latent space. With the introduction of this latent task space, we decouple the problem of learning the task manifold from the problem of inducing the curriculum. Additionally, the adversary can be trained using single-step RL unlike task-space UED teachers and thus can avoid the long-horizon credit assignment problem. Also, the same CLUTR teacher architecture can be used for different simulators unlike PAIRED. For the latent task space, we train an LSTM based recurrent Variational Auto-Encoder with data randomly sampled from the parameter space in an unsupervised manner, without requiring any costly interaction with the simulator. We solve the combinatorial explosion problem faced by the parameter-space UED teachers by curating our training dataset. For example, we sort the obstacle locations for navigation task and consequently each grid configuration is represented uniquely in the dataset. Moreover, the theoretical properties and guarantees provided by PAIRED are agnostic of the adversary model class and we have chosen a model class that can represent any adversary, all such theoretical properties of PAIRED holds for CLUTR seamlessly. Our experimental results show CLUTR outperforms PAIRED both, in terms of generalization and sample-efficiency, in pixel-based continuous Car Racing and partially observable discrete navigation tasks.

In summary, we make the following contributions: i) we introduce CLUTR, a novel unsupervised curriculum learning algorithm by augmenting existing UED teachers with unsupervised task-representation learning, holding the theoretical properties by PAIRED. ii) Latent space UED decouples Task Representation Learning from the Curriculum Generation Objective. The Latent-space teacher can be trained using single-step RL to avoid the long-horizon credit assignment problem. The latent space, by construction, also solves the combinatorial explosion problem faced by parameter-space UED teachers. iii) Our experimental results show CLUTR outperforms PAIRED both, in terms of generalization and sample-efficiency, in MiniGrid and CarRacing environments.

2 BACKGROUND

2.1 UNSUPERVISED ENVIRONMENT DESIGN (UED)

As introduced by Dennis et al. (2020) UED is the problem of inducing a curriculum by designing a distribution of concrete fully-specified environments from an underspecified environment with free parameters. The fully specified environments are represented using a Partially Observable Markov Decision Process (POMDP) represented by $(A, O, S, \mathcal{T}, \mathcal{I}, \mathcal{R}, \gamma)$, where A, O, and S denote the action, observation, and state spaces, respectively. $\mathcal{I} \rightarrow O$ is the observation function, $\mathcal{R} : S \rightarrow \mathbb{R}$ is the reward function, $\mathcal{T} : S \times A \rightarrow \Delta(S)$ is the transition function and γ is the discount factor. The

underspecified environments are defined in terms of an Underspecified Partially Observable Markov Decision Process (UPOMDP) represented by the tuple $\mathcal{M} = (A, O, \Theta, S^{\mathcal{M}}, \mathcal{T}^{\mathcal{M}}, \mathcal{I}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, \gamma)$. Θ is a set representing the free parameters of the environment and is incorporated in the transition function as $\mathcal{T}^{\mathcal{M}} : S \times A \times \Theta \to \Delta(S)$. Traditionally (e.g., in Dennis et al. (2020) and Jiang et al. (2021a)) Θ is considered as a trajectory of environment parameters $\vec{\theta}$ or just θ. For example θ can be a sequence of integers denoting obstacle locations for a gridworld nagivation problem. We denote a concrete environment generated with the parameter $\vec{\theta} \in \Theta$ as $\mathcal{M}_{\vec{\theta}}$ or simply $\mathcal{M}_\theta$. The value of a policy π in $\mathcal{M}_\theta$ is defined as $V^\theta(\pi) = \mathbb{E}[\sum_{t=0}^{T} r_t \gamma^t]$, where r_t is the discounted reward obtained by π in $\mathcal{M}_\theta$.

2.2 PAIRED

PAIRED Dennis et al. (2020) solves UED with adversarial training with a game involving three players: the agent, which they call the protagonist, π_P and an antagonist π_A are trained in tasks generated by the teacher $\tilde{\theta}$. PAIRED objective is to maximize $U(\pi_P, \pi_A, \tilde{\theta}) = \mathbb{E}_{\theta \sim \tilde{\theta}}[\textsc{Regret}^\theta(\pi_P, \pi_A)]$. Regret is defined by the difference of the discounted rewards obtained by the antagonist and the agent in the generated tasks, i.e., $\textsc{Regret}^\theta(\pi_P, \pi_A) = V^\theta(\pi_A) - V^\theta(\pi_P)$. The teacher agent is trained with an RL algorithm with U as the reward while, the protagonist and antagonist agents are trained using usual discounted rewards from the environments.

3 Curriculum Learning via Unsupervised Task Representation Learning

In this section we formally define Unsupervised Latent Environment Design and how CLUTR as an instance of it, followed by the outline and further details of CLUTR.

3.1 Graphical Formulation of CLUTR

We use a hierarchical graphical model to formulate the latent environment design problem. Let's assume that R is a random variable that denotes a measure of success defined using the protagonist and antagonist agents and z be a latent random variable. We use the following graphical model: $z \to E \to R$ where z generates an environment E and R is the success defined over E. Both E and R are observed while z is an unobserved. R covers a broad range of measures used in different UED methods including PAIRED and DR (Domain Randomization). In PAIRED, R represents the Regret as the difference of returns between the antagonist and protagonist agents and it depends on the environments that the agents are evaluated on.

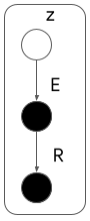


Figure 1: Hierarchical Graphical Model for CLUTR

We use a variational formulation of UED by using the above graphical model. We first define the variational objective as the KL-divergence between an approximate posterior distribution and true posterior distribution over latent variable z,

$$\begin{aligned} D_{KL}(q(z)|p(z|R,E)) &= E_{z\sim q(z)}[logq(z)] - E_{z\sim q(z)}[logp(z|R,E)] \\ &= E_{z\sim q(z)}[logq(z)] - E_{z\sim q(z)}[logp(R,E,z)] + logp(R,E) \end{aligned}$$

where both R and E are given.

Next, we write the ELBO,

$$\begin{aligned}
ELBO &= E_{z\sim q(z)}[logq(z)] - E_{z\sim q(z)}[logp(R,E,z)] \\
&= E_{z\sim q(z)}[logq(z)] - E_{z\sim q(z)}[logp(R|E)p(E|z)p(z)] \\
&= E_{z\sim q(z)}[logq(z)] - E_{z\sim q(z)}[logp(z)] - E_{z\sim q(z)}[logp(E|z)] - E_{z\sim q(z)}[logp(R|E)] \\
&= E_{z\sim q(z)}[log\frac{q(z)}{p(z)}] - E_{z\sim q(z)}[logp(E|z)] - logp(R|E) \\
&= D_{KL}(q(z)|p(z)) - E_{z\sim q(z)}[logp(E|z)] - logp(R|E) \\
&= VAE(z,E) - logp(R|E)
\end{aligned}$$

The above ELBO lends itself nicely in a pretraining objective for curriculum learning – optimize the VAE to learn unsupervised task representations and fine-tune this pretrained model to optimize ELBO.

We can also induce an objective that includes minimax REGRET. Let R be distributed according to an exponential distribution, $p(R|E) \propto exp(\text{REGRET}(\pi_P, \pi_A|E))$,

we derive,

$$ELBO \approx VAE(z,E) - \text{REGRET}(R,E)$$

where the normalizing factor is ignored.

Note that, $\text{REGRET}(R,E) = \text{REGRET}^{\theta}(\pi_P, \pi_A)$ where, θ denotes the free parameters of the environment E and is given by the VAE decoder function $\mathcal{G} : \mathcal{Z} \to \Theta$. We define CLUTR teacher policy as $\Lambda : \Pi \to \Delta(\mathcal{Z})$, where Π is a set of possible agent policies and $\mathcal{Z}$ is as the latent space.

3.2 CLUTR

Now we define the outline of the CLUTR algorithm (Algorithm 1). The main difference between PAIRED and CLUTR algorithmic outline is, instead of generating the parameter vectors directly, CLUTR teacher generates the latent task vector z and it is translated to the task-parameter θ using the decoder $\mathcal{G}$. For this reason, CLUTR teachers can be trained using single-step RL. In PAIRED the teacher generates one parameter value at a time and is conditioned with the current partially-generated task. For some simulators, it might not be possible get the state of a partially generated task. As CLUTR does not condition on the partially generated tasks, it doed not require this assumption on the simulators. Furthermore, PAIRED requires different teacher architecture for different simulators, depending on the state-space of the underlying POMDP. CLUTR teacher architecture, on the other hand, is agnostic of the POMDP/UPOMDP it is solving, hence the same architecture can be used across different problems.

Algorithm 1 CLUTR

Pretrain VAE with randomly sampled tasks from Θ
Randomly initialize Agent π^P, Antagonist π^A, and Adversary $\tilde{\Lambda}$;
while *not converged* **do**
  Use adversary to sample latent task vector: $z \sim \mathcal{Z}$
  Create POMDP M_θ where $\theta = \mathcal{G}(z)$ and $\mathcal{G}$ is the VAE decoder function
  Collect Agent trajectory τ^P in M_θ. Compute: $U^\theta(\pi^P) = \sum_{i=0}^{T} r_t\gamma^t$
  Collect Antagonist trajectory τ^A in M_θ. Compute: $U^\theta(\pi^A) = \sum_{i=0}^{T} r_t\gamma^t$
  Compute: $\text{REGRET}^\theta(\pi^P, \pi^A) = U^\theta(\pi^A) - U^\theta(\pi^P)$
  Train Protagonist policy π^P with RL update and reward $R(\tau^P) = U^\theta(\pi^P)$
  Train Antagonist policy π^A with RL update and reward $R(\tau^A) = U^\theta(\pi^A)$
  Train Adversary policy $\tilde{\Lambda}$ with RL update and reward $R(\tau^{\tilde{\Lambda}}) = \text{REGRET}$
end while

3.3 UNSUPERVISED LATENT TASK REPRESENTATION LEARNING

We learn the latent task representation without any costly interaction with the simulator. Aligning with Dennis et al. (2020) and Jiang et al. (2021a), we represent the parameters with a sequence of integers. We use the LSTM based VAE architecture from Bowman et al. (2015), used to generate continuous representations of sentences. Even though the parameters are integer valued, we consider each number as word-tokens for generality and we use random word-embedding. The training data is generated with random tasks by sampling the underspecified specification. Further details can be found in the experiments section.

4 EXPERIMENTS

In this section we first assess the empirical performance of CLUTR in comparison with PAIRED in two distinct set of tasks: i) Pixel-Based Car Racing with continuous control and dense rewards and ii) Partially Observable Navigation Tasks with discrete control and sparse rewards. We then compare CLUTR and PAIRED induced curricula in terms of regret, agent returns, and emergent complexity. Furthermore, we see the impact of sorting the VAE training data on CLUTR.

All the agents are trained with PPO(Schulman et al. (2017)). Full details of the environments, network architectures of the teacher and student agents, the VAE, the training hyperparameters are available in the Appendix.

4.1 PERFORMANCE ON PIXEL-BASED CONTINUOUS CONTROL CARRACING ENVIRONMENT

We evaluate both CLUTR and PAIRED on the CarRacing environment, originally proposed by OpenAI Gym Brockman et al. (2016), and later reparameterized by Jiang et al. (2021a) with Bézier Curves Mortenson (1999) for UED algorithms. This environment requires the agents to drive a full lap around a track defined by a Bézier Curve—specified with a sequence of upto 12 arbitrary control points, uniquely defining a unique track subject to predefined curvature constraints Jiang et al. (2021a). The environment is a continuous control with partially observable pixel observations, three dimensional continuous action space, and dense rewards. We train both PAIRED and CLUTR with 5M timesteps aligning with Jiang et al. (2021a). To train the VAE we generate 1 Million sorted sequences of integer parameters of length 12 denoting the control points similar to Jiang et al. (2021a) and train a VAE for 1 Million training steps. Note that, the sequences can have duplicates integers.

To test the Zero-shot generalization on the Car Racing tasks, we evaluate both CLUTR and PAIRED agents on 20 levels replicating official human-designed real Formula One (F1) tracks, also obtained from Jiang et al. (2021a). CLUTR outperforms PAIRED in all 20 test-tracks as seen in Figure **??**. Note that these tracks cannot be generated with just 12 control points and hence are significantly OOD than any tracks that the CLUTR and PAIRED teachers could generate during training. PAIRED struggles to train a robust agent within the allotted training budget. As mentioned in Jiang et al. (2021a) PAIRED overexploits the relative strengths of the antagonist over the protagonist and generates a curriculum that gradually reduces the level complexity. However, CLUTR overcomes this and generates a curriculum where antagonist and protagonist closely compete with each other as seen in Figure 5b and shows a robust generalization on the unseen F1 benchmark. Figure 3b compares sample efficiency of CLUTR with PAIRED by evaluating the agents on four selected tracks: Vanilla, Singapore, Germany, Italy during training. It can be seen that CLUTR outperforms PAIRED significantly. These test environments were used in anyway neither during training CLUTR(or, PAIRED) now while designing it.

4.2 PERFORMANCE ON PARTIALLY OBSERVABLE NAVIGATION TASKS

The navigation tasks are performed on the popular MiniGrid environment, originally introduced by Chevalier-Boisvert et al. (2018) and adopted by Dennis et al. (2020) for UEDs. For these tasks, an agent explores a grid world to find a goal while navigating around obstacles. The environment is partially observable, hence we use the recurrent neural network (RNN) architecture used in Dennis et al. (2020) to represent the student policies. Each navigation task is represented with a sequence of integers denoting the locations of the obstacles, the goal, and the starting position of the agent: on

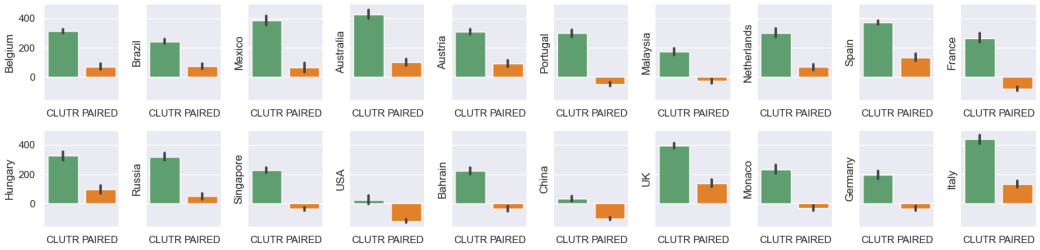
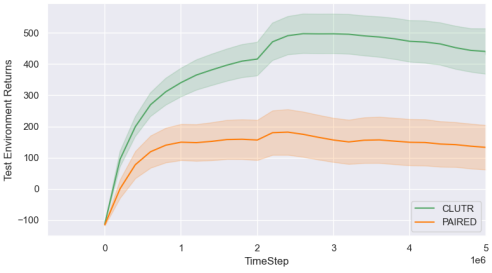
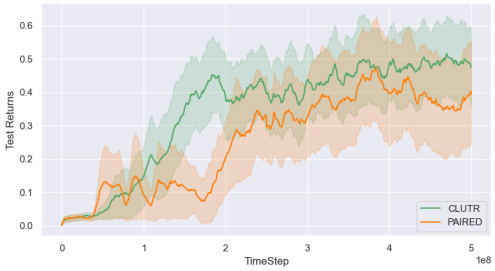


Figure 2: Zero-shot generalization of both PAIRED and CLUTR agents after 5M timesteps on the full F1 benchmark. CLUTR outperforms PAIRED on every track. For each track, we test the agents on 10 different episodes and the error bar denotes the standard error.



(a) Test Returns on Selected Tracks: Vanilla, Singapore, Germany, and Italy



(b) Test Returns on seven selected navigation tasks

Figure 3: Comparison of PAIRED and CLUTR performance on selected tasks during training. In both tasks CLUTR show much better Sample Efficiency (as well as performance) the PAIRED.

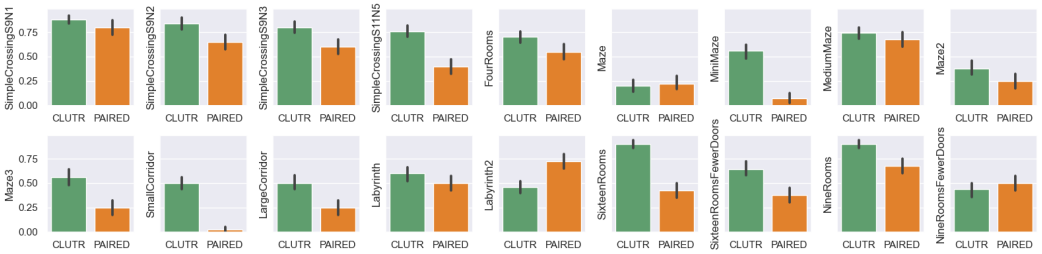


Figure 4: Zero-shot generalization, in terms of percent of environments solved, of both PAIRED and CLUTR agents after 500M timesteps on the 18 navigation tasks. CLUTR outperforms PAIRED on 15 out of the 18 tasks, and is comparable in one of the rests. For each of these tasks, we test the agents on 100 different episodes and the error bar denotes the standard error.

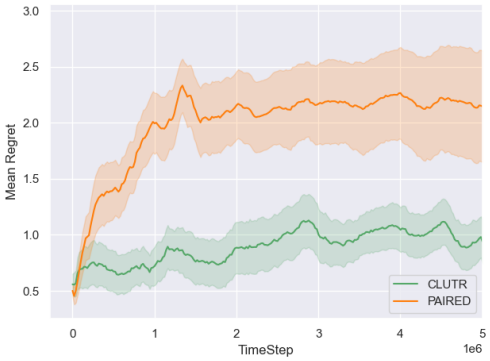
a 15 by 15 grid similar to Dennis et al. (2020). PAIRED teacher outputs a sequence of 52 integers, while the first 50 numbers denote the location of obstacles and may contain duplicates to allow generation of tasks with fewer than 50 obstacles. To train CLUTR VAE, we generate 1 Million random grids, with the obstacle locations sorted, and the number of obstacles uniformly varying from zero to 50. Note that the results reported in Dennis et al. (2020) is obtained after 3 Billion timesteps of training, while we run both PAIRED and CLUTR for 500M timesteps due to the huge computational and time resource needed to run a training with 3 Billion timesteps.

We evaluate both CLUTR and PAIRED trained agents on 18 unseen navigation tasks from Dennis et al. (2020)and Jiang et al. (2021a). Figure 4 shows, CLUTR outperforms PAIRED in 15 out of the 18 grids and shows comparable performance in one of the rests. Figure 3a shows performance on 7 selected grids [2] during training. It shows CLUTR reaches a better performance much faster than PAIRED—roughly the gap between the rough peaks of the curves are around 200M timesteps apart.

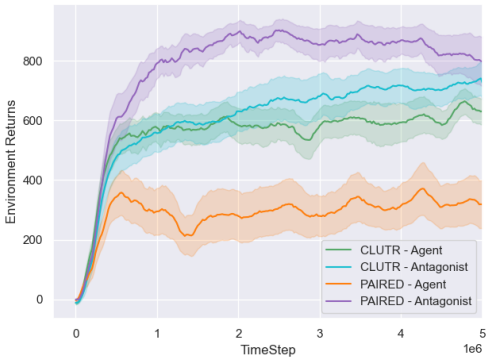
4.3 Analysis of the training metrics: Regret and Agent Returns

Figure 5 shows the mean regret and agent returns on the teacher generated tasks. CLUTR shows significantly lower regret throughout the training compared to PAIRED, which means the agent and the antagonist obtain similar returns while antagonist being slightly better then the agent resulting in the positive regret (see Figure 5a and Figure 5c). For the navigation task, PAIRED regret converges at around 250M timesteps, which is around twice in magnitude with respect to the CLUTR regret value. From a curriculum learning perspective, we want to train the agent on tasks which are slightly harder than it can already solve or, on the tasks the agent can solve them already but can obtain better returns. In practice, both the agent and the antagonist are trained in the same training context e.g., same hyper-parameters, model-architecture, and training environments, differing only by their random initial weights. Hence, a low regret means that the teacher is generating tasks that are either slightly harder then the tasks the agent can solve now (because the antagonist is solving them) or, tasks in which the antagonist is performing slightly better than the agent. Hence, the tasks are at the agent's 'frontier of capability'. Figure 5b and Figure 5d shows the agent and antogonist returns on the UED generated environment, which shows CLUTR has a smaller gap between its agent and antagonist returns than PAIRED. The regret and return curves also show that CLUTR and PAIRED—both of which are optimized by the same minimax-regret criteria and differs only by the adversary teacher—show similar convergence pattern, while CLUTR is doing it sooner, more smoothly, and converging to a better local optima of regret. These results signifies that having a pretrained task representation makes the job of teacher easier, possibly by the regularization provided fixed task representations by our pretrained decoder.

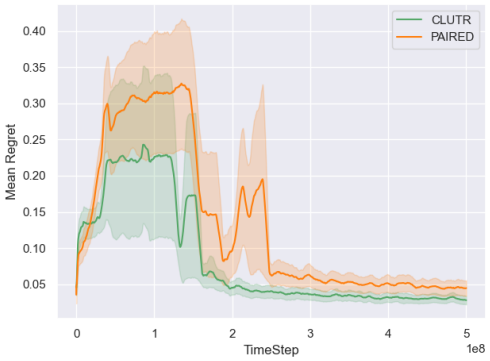
[2] Sixteen Rooms, Sixteen Rooms with Fewer Doors, Labyrinth, Labyrinth2, Maze, Maze2, and Large Corridor



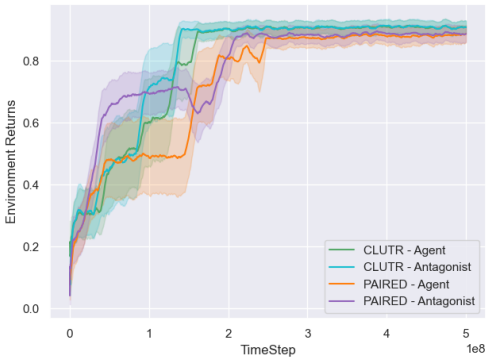
(a) Mean Regret - Car Racing



(b) Returns on UED generated Car Racing tracks



(c) Mean Regret - Navigation



(d) Returns from UED generated grids

Figure 5: Mean Regret and agent returns during training. CLUTR shows a smaller regret value indicating a better UED curriculum.

5 Related Work

PAIRED (Dennis et al. (2020)) was the first to introduce a minimax regret based teacher algorithm for Unsupervised Curriculum Learning/Environment Design. The widely popular Domain Randomization (Jakobi (1997), Sadeghi & Levine (2016), Tobin et al. (2017)) is another instance of UED methods, where the teacher follows an uniform random strategy. Another common form of UED teachers (Morimoto & Doya (2005), Pinto et al. (2017)) follow minimax training, i.e., generating tasks that to minimize the return of the current agent. Dennis et al. (2020) shows that training a teacher under minimax regret criteria can generate feasible tasks with emergent complexity unlike Domain Randomization and Minimax training. CLUTR teacher which is also trained with minimax regret holds the same theoretical guarantees as PAIRED as no further assumptions were made on the class of teachers we can represent.

Attempts have also been made to improve PAIRED by augmenting it with other techniques. Jiang et al. (2021a) introduces REPAIRED that combines PAIRED with level (or, task according to the naming convention used in this paper) replay capabilities, i.e., Prioritized Level Replay Jiang et al. (2021b) that selectively samples previously generated training tasks. PLR estimates learning potentials for each previously generated tasks and the utility of replaying the previous levels. REPAIRED, by combining both PAIRED and PLR, co-evolves two teachers: one who generates new environments where the other selects a previously generated task for replay. Jiang et al. (2021a) also combined Domain Randomization with PLR which outperformed REPAIRED and PAIRED in both Car Racing and MiniGrid Navigation tasks.

6 Conclusion and Future Work

In this work, we propose CLUTR, an unsupervised environment design method via unsupervised task representation learning. By augmenting PAIRED with a latent task space, it shows similar theoretical properties and guarantees as PAIRED. However, unlike PAIRED which builds the tasks from scratch one parameter at a time, CLUTR 'chooses' tasks from the latent space. CLUTR poses several advantages over PAIRED. PAIRED simultaneously learns a task manifold as well as how to navigate it to create an effective curriculum, which is difficult due to the task manifold evolves non-stationarily over time. CLUTR, by introducing the latent task-space, decouples learning task representations from curriculum objectives. Secondly, it is possible to train CLUTR teacher with single-step RL avoiding long-horizon credit assignment. Additionally, CLUTR teacher does not face the combinatorial explosion problem faced by PAIRED teacher or other parameter-space teacher agents. Furthermore, CLUTR teacher architecture is agnostic of the problem, unlike PAIRED. The task-manifold is learned with a recurrent LSTM VAE from random tasks. Our experiments show CLUTR outperforms PAIRED in Car Racing and MiniGrid environments both in terms of sample efficiency and generalization.

References

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

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

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for gymnasium. `https://github.com/Farama-Foundation/MiniGrid`, 2018.

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.

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

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.

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, 2015.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

Michael E Mortenson. *Mathematics for computer graphics applications*. Industrial Press Inc., 1999.

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.

Lerrel Pinto, James Davidson, and Abhinav Gupta. Supervision via competition: Robot adversaries for learning tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1601–1608. IEEE, 2017.

Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image, 2016. URL https://arxiv.org/abs/1611.04201.

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

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, 2016.

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.