SUPPLEMENTARY MATERIALS FOR CLUTR: CURRICULUM LEARNING VIA UNSUPERVISED TASK REPRESENTATION LEARNING

A ADDITIONAL DETAILS OF CLUTR

A.1 CLUTR OBJECTIVE DERIVATION

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 graphical model in Figure-8 where z generates an environment E and R is the success defined over E. Both E and R are observed variables while z is an unobserved variable. 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.

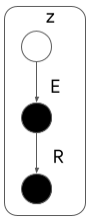


Figure 8: Hierarchical Graphical Model for CLUTR

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,

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

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.

A.2 ROBUSTNESS GUARANTEES

CLUTR essentially proposes including a pretrained latent space within the teacher/generator. From the teacher's perspective, the difference is while the PAIRED teacher starts from randomly initialized weights, CLUTR starts from the pretrained weights. Thus, CLUTR does not impose new assumptions on possible teacher policies. Furthermore, CLUTR does not change any other specifics of the underlying PAIRED algorithm. Hence, CLUTR holds the same theoretical robustness guarantees provided by PAIRED.

In practice, both CLUTR and PAIRED deviate from these theoretical guarantees. For example, both algorithms approximate the regret value, which is the case for other regret-based UEDs such as Robust PLR and REPAIRED (Jiang et al. (2021a)). Also, the robustness guarantee depends on reaching the Nash equilibrium of the multiagent adversarial game. However, gradient-based multi-agent RL has no convergence guarantees and often fails to converge in practice(Mazumdar et al. (2019)). We also note that, by introducing the latent space, CLUTR VAE might not have access to the full task space due to practical limitations on training, e.g., the training dataset not having all possible tasks. However, when the decoder is allowed to be finetuned, CLUTR will have access to the full task space, similar to PAIRED. Our empirical results (discussed in Section 5.4) suggest that keeping the pretrained decoder fixed performs better than finetuning it, so we kept it fixed for our main experiments. We also want to mention that we used the flexible regret objective for CarRacing in Section 5.1. When the flexible objective is used, CLUTR (and PAIRED) might not hold the robustness guarantee as it changes the dynamics of the underlying game between the teacher and the agents. However, we also experimented with the standard regret objective and obtained better performance than PAIRED as discussed in Section C.2.

B Training Details

B.1 Environment Details

Car Racing: The CarRacing environment was originally proposed by OpenAI Gym Brockman et al. (2016), and later has been 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 closed-loop track. The track is defined by a Bézier Curve modeled with a sequence of upto 12 arbitrary control points, each spaced within a fixed radius $B/2$ of the center of the $B \times B$ field. This sequence of control points can uniquely identify a track, subject to a set of predefined curvature constraints Jiang et al. (2021a). The control points are encoded in a 10×10 grid—a discrete down-sampled version of the racing track field. Each control point hence is a integer denoting a cell of the grid and the cell coordinates are upscaled to match the original scale of the field afterwards. This ensures no two control points are too close together, preventing areas of excessive track overlapping. The track consists of a sequence of L polygons and the agent receives a reward of $1000/L$ upon visiting each unvisited polygon and a penalty of -0.1 at each time step to incentivize completing the tracks faster. Episodes terminate if the agent drives too far off-track but is not given any additional penalty. The agent controls a 3 dimensional continuous action space corresponding to the car's steer: torque $\in [-1.0, 1.0]$, gas: acceleration $\in [0, 0, 1.0]$, and brake: deceleration $\in [0.0, 1.0]$. Each action is repeated 8 times. The agent receive a $96 \times 96 \times 3$ RGB pixel observation. The top 84×96 portion of the frame contains a clipped, egocentric, bird's eye view of the horizontally centered car. The bottom 12×96 segment simulates a dashboard visualizing the agent's latest action and return. Snapshots of the test track in the F1 benchmark are shown in Figure 9.

Minigrid: The environment is partially observable and based on Chevalier-Boisvert et al. (2018) and adopted for UED by Dennis et al. (2020). 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 a 15×15 grid similar to Dennis et al. (2020). The grids are surrounded by walls on the sides, making it essentially a 13×13 grid. Dennis et al. (2020) parameterizes the locations using integers. Each task is a sequence of 52 integers, while the first 50 numbers denote the location of obstacles followed by the goal and the agent's initial location. The sequences may contain duplicates to allow the generation of navigation tasks with fewer than 50 obstacles. Snapshots of the test grids used in our paper are shown in Figure 10.

B.2 Network Architectures

All the student and teacher agents are trained with PPO Schulman et al. (2017).

Student Architecture

For CarRacing, we use the same student architecture as Jiang et al. (2021a). The architecture consists an image embedding module composed of 2D Convolutions with square kernels of sizes 2,2,2,2,3,3, stride lengths 2,2,2,2,1,1 and channel outputs of 8, 16, 64, 128, 256 stacked together. The image embedding is of size 256 and is passed through a Fully Connected (FC) layer of 100 hidden units

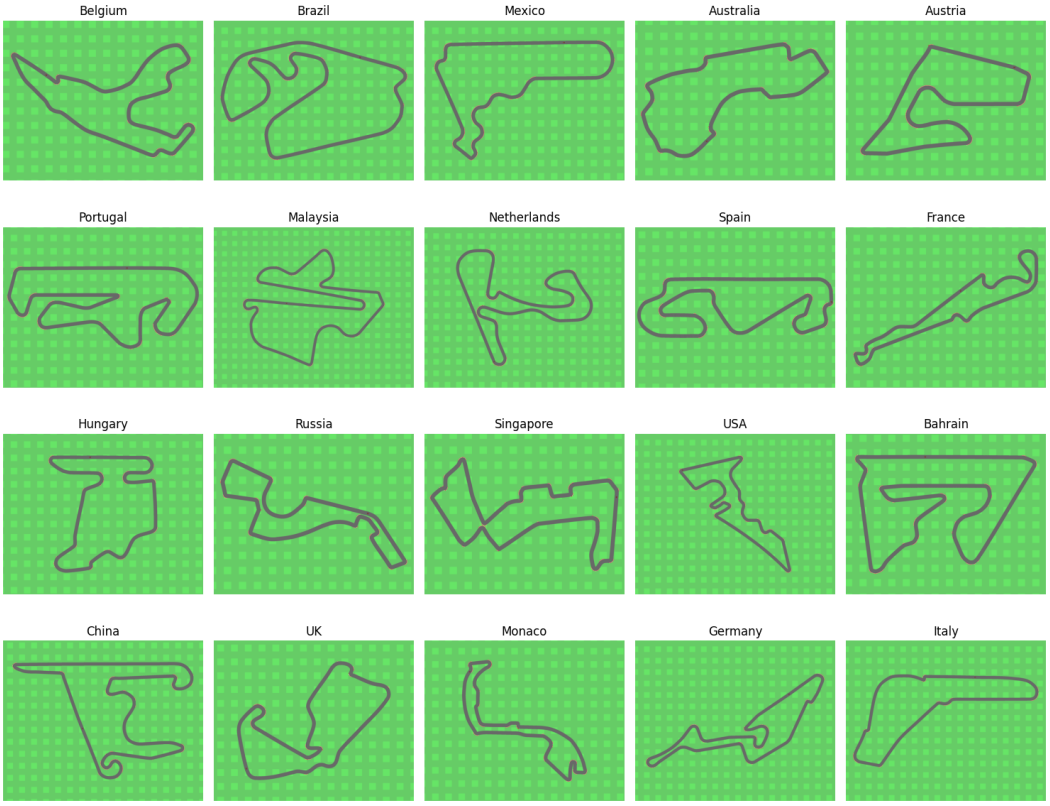


Figure 9: Snapshots of the test tracks in F1 benchmark

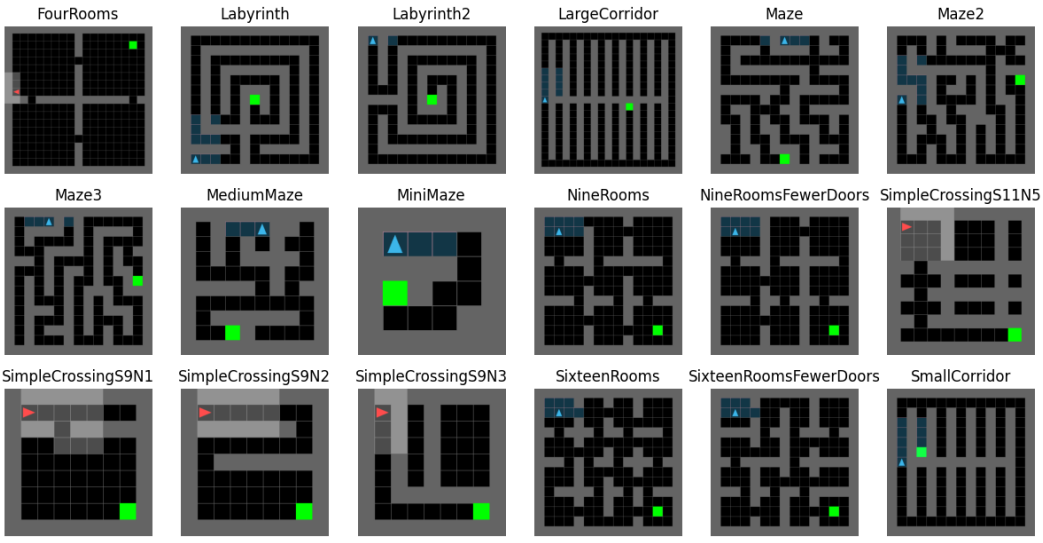


Figure 10: Snapshots of the test grids for MiniGrid

and then passed through ReLU activations. This embedding is then passed through two FC with 100 hidden neurons, and then a softplus layer, and finally added to 1 for the beta distribution used for the continuous action space. Further details can be found in Jiang et al. (2021a).

For navigation tasks, we use the same student architecture as Dennis et al. (2020). The observation is a tuple with a $5 \times 5 \times 3$ grid observation and a direction integer in $[0-3]$. The grid view is fed to a convolutional layer with kernels of size 3 with 16 filters and the direction integer is passed through a FC with 5 units. This is followed by an LSTM of size 256, and then to two FC layers with 32 units, which connect to the policy outputs. The value network uses the same architecture.

Teacher Architecture

For CarRacing, CLUTR teacher takes a random noise and generates a continuous vector, i.e., the latent task vector. We pass the random noise through a feed-forward network with one hidden layer of 8 neurons as the teacher. The output of this layer is fed through two separate fully-connected layers, each with a hidden size of 100 and an output dimension equal to the latent space dimension, followed by soft plus activations. We then add 1 to each component of these two output vectors, which serve as the α and β parameters respectively for the Beta distributions used to sample each latent dimension. In our experiments, we used a 64-dimensional latent task space. For Minigrid experiments, we use a network architecture similar to Dennis et al. (2020) but take only the random noise as input. The adversary network generates discrete actions, but we map them to real numbers to feed into the VAE decoder.

VAE architecture

We use the architecture proposed in Bowman et al. (2015). We use a word-embedding layer of size 300 with random initialization. The encoder comprises a conditional 'Highway' network followed by an LSTM. The Highway network is a two-staged network stacked on top of each other. Each stage computes $\sigma(x) \odot f(G(x)) + (1 - \sigma(x)) \odot Q(x)$, where x is the inputs to each of the highway network stages, G and Q is affine transformation, $\sigma(x)$ is a sigmoid non-linearization, and $\odot$ is element-wise multiplication. G and Q are feed-forward networks with a single hidden layer with equal input and output dimensions of 300, equal to the word-embedding output dimension. We use ReLU activation as f. The highway network is followed by a bidirectional LSTM with a single layer of 600 units. The LSTM outputs are passed through linear layer of dimension 64 to get the VAE mean and log variance. The mean vectors are passed through a hyperbolic tangent activation and for the navigation tasks linearly scaled in $[-4, 4]$. The decoder takes in latent vectors of dimension 64 and passes through a bidirectional LSTM with two hidden layers of size 800 and follows it by a linear layer with size equaling the parameter vector dimension.

B.3 HYPERPARAMETERS

All our agents are trained with PPO (Schulman et al. (2017)). We did not perform any hyperparameter search for our experiments. The CarRacing experiments used the same parameters used in Jiang et al. (2021a), and the Minigrid experiments used the parameters from Dennis et al. (2020). VAE was trained on the parameters from Bowman et al. (2015). The detailed parameters are listed in Table 2 and Table 3.

Parameter	Value
Batch Size	32
Number of Training Steps	1000000
Reconstruction Weight	79
Latent Variable Size	64
Word Embedding size	300
Maximum Sequence Length	52
Encoder Activation	Hyperbolic Tangent
Learning Rate	0.00005
Dropout	0.3

Table 2: Hyperparameters for training the Task VAE

Parameter	CarRacing	MiniGrid
γ	0.99	0.995
λ_{GAE}	0.9	0.95
PPO rollout length	125	256
PPO epochs	8	5
PPO minibatches per epoch	4	1
PPO clip range	0.2	0.2
PPO number of workers	16	32
Adam learning rate	3e-4	1e-4
Adam ϵ	1e-5	1e-5
PPO max gradient norm	0.5	0.5
PPO value clipping	no	yes
Return normalization	yes	no
Value loss coefficient	0.5	0.5
Student entropy coefficient	0	0
Action Repeat	8	-

Table 3: Hyperparameters for PAIRED and CLUTR PPO training.

B.4 VAE Training Data

For CarRacing, we follow the same parameterization as Jiang et al. (2021a): each track is defined with a sequence of up to 12 integers denoting control points of a Bézier Curve. . Each control point is represented with an integer. We generate 1M random sorted integer sequences of fixed length 12 with duplicates—which enables generating tracks defined with less than 12 control points. For navigation tasks we use the parameterization of Dennis et al. (2020), generating upto 50 obstacles for each task for a 15×15 grid, surrounded by walls, effectively an active area of 13×13. Hence, each location is numbered in 1 to 169. Every number except the last two of the sequence represent obstacle locations, and the last two for the goal and agent location, respectively. The parameter vector is thus partially permutation invariant. We uniformly generate 1M sequences of variable length between 2 and 52 (inclusive). The obstacle locations are sorted.

C Detailed Results on CarRacing

C.1 Detailed Comparison on Full F1 Dataset

We used the flexible regret approximation for the results presented in the main paper. The flexible regret objective is a more robust variant of the standard regret estimation (both introduced in Dennis et al. (2020)). It is defined by the difference between the average score of the agent and antagonist returns and the score of the policy that achieved the highest average return. Thus, the flexible objective blurs the distinction between the agent and the antagonist. Hence we designate the agent achieving the higher average training return during the last 10 steps as the agent.

Figure 11 compares how different UEDs perform during training by periodically evaluating them on Four Selected Tracks: Vanilla, Singapore, Germany, and Italy. These tracks were selected aligning with Jiang et al. (2021a). Based on these selected tracks, CLUTR performance plateaus around 2.5M timesteps. Robust PLR starts slowly but surpasses all the other methods after 5M timesteps.

Table 4 shows the comparison between the final agents trained with CLUTR and other UED algorithms. It is to be noted that, all the UED methods except CLUTR was trained for 5M timesteps where CLUTR was run for 2M timesteps. CLUTR outperforms PAIRED by a big margin with 18x bigger mean return on the entire F1 Dataset. CLUTR also outperforms Domain Randomization, PLR, and REPAIRED and only falls short to Robust PLR. Nonetheless, CLUTR shows competitive results compared to Robust PLR, showing comparable results in seven out of the 20 test tracks and outperforming in the Netherlands track. CLUTR also outperforms the non-UED SOTA on the full F1 dataset. CLUTR outperforms the Attention Agent on 9 out of the 20 tracks and shows comparable performance in one.

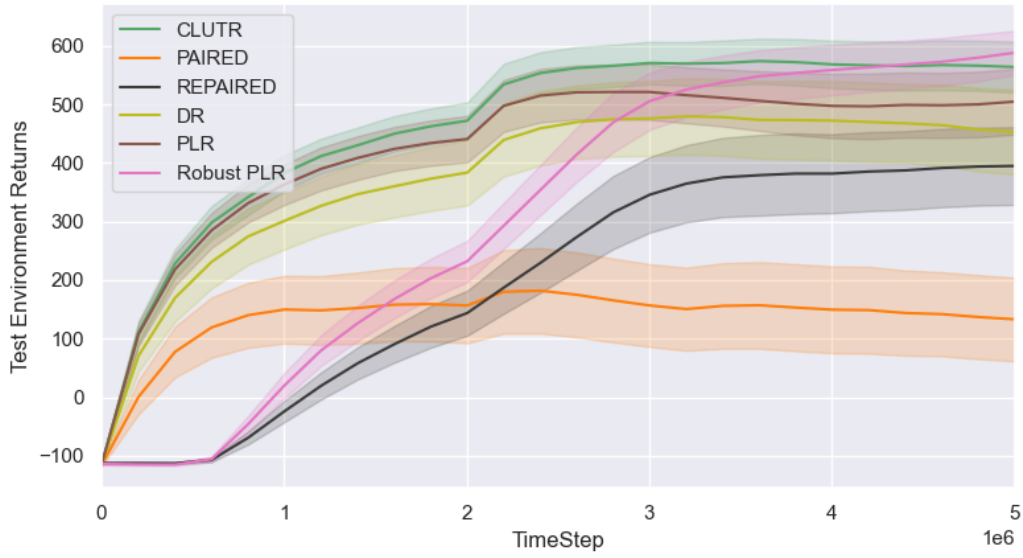


Figure 11: Comparison of mean agent returns on Four Selected Tracks: Vanilla, Singapore, Germany, and Italy. Based on these selected tracks, CLUTR improves a bit after 2M timesteps later the performance plateaus. Robust PLR starts slowly but surpasses all the other methods after 5M timesteps.

C.2 CLUTR WITH STANDARD REGRET LOSS

We also train CLUTR with the standard regret loss for 5M timesteps. Figure 12 compares the impact of standard/flexible regret loss on the regret and agent returns during training. With standard regret loss, CLUTR shows a lower regret value, but shows similar pattern. The CLUTR agent achieves better returns with flexible loss throughout the training.

Figure 13 compares the mean regret and agent training returns with PAIRED. CLUTR with standard loss shows much lower regret than PAIRED (Figure 13a). Figure 13b shows that the CLUTR agents compete closely, while PAIRED antagonist achieves much higher returns than the PAIRED agent which leads to higher regret returns for the teacher agent but results in a weak student agent. To test the Zero-shot generalization, we evaluate CLUTR with the standard loss on the full F1 benchmark. Figure 14 shows CLUTR with standard regret loss outperforms PAIRED in all the 20 test tracks. This implies that CLUTR outperforms PAIRED irrespective of the choice of the loss function (standard/flexible). Figure 15 compares the sample efficiency of CLUTR with the standard regret loss with PAIRED by evaluating the agents on four selected tracks (Vanilla, Singapore, Germany, Italy) during training. It can be seen that CLUTR, even without the regret loss, outperforms PAIRED significantly. We note that these test environments were not used in any way, neither during training CLUTR (and PAIRED) nor while designing it.

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 task complexity. However, CLUTR overcomes this and generates a curriculum where the agent and the antagonist closely compete (Figure 13b) and shows a robust generalization on the unseen F1 benchmark.

C.3 EXTENDED ANALYSIS ON IMPACT OF SORTING TRAINING DATA FOR VAE TRAINING

The non-sorted dataset was generated by shuffling each track of the original VAE training dataset 10 different times, resulting in a 10X bigger dataset (10M tracks). It was trained for 5X longer for 5M training steps. We planned on training for 10M gradient steps (10X than the original VAE) but stopped at 5M as it converged much sooner. We ran both CLUTR and CLUTR-shuffled, i.e.,

Track	DR	PLR	Robust PLR	PAIRED	REPAIRED	CLUTR (2M)	Attention Agent
Australia	484 ± 29	545 ± 23	**692 ± 15**	100 ± 22	414 ± 27	**683 ± 20**	826
Austria	409 ± 21	442 ± 18	**615 ± 13**	92 ± 24	345 ± 19	507 ± 19	*511*
Bahrain	298 ± 27	411 ± 22	**590 ± 15**	-35 ± 19	295 ± 23	414 ± 20	*372*
Belgium	328 ± 16	327 ± 15	**474 ± 12**	72 ± 20	293 ± 19	429 ± 15	668
Brazil	309 ± 23	387 ± 17	**455 ± 13**	76 ± 18	256 ± 19	363 ± 18	*145*
China	115 ± 24	84 ± 20	**228 ± 24**	-101 ± 9	7 ± 18	**254 ± 28**	344
France	279 ± 32	290 ± 35	**478 ± 22**	-81 ± 13	240 ± 29	**498 ± 31**	*153*
Germany	274 ± 23	388 ± 20	**499 ± 18**	-33 ± 16	272 ± 22	404 ± 20	*214*
Hungary	465 ± 32	533 ± 26	**708 ± 17**	98 ± 29	414 ± 29	630 ± 24	769
Italy	461 ± 27	588 ± 20	**625 ± 12**	132 ± 24	371 ± 25	**639 ± 16**	798
Malaysia	236 ± 25	283 ± 20	**400 ± 18**	-26 ± 17	200 ± 17	**426 ± 22**	*300*
Mexico	458 ± 33	561 ± 21	**712 ± 12**	67 ± 31	415 ± 30	627 ± 19	*580*
Monaco	268 ± 28	360 ± 32	**486 ± 19**	-28 ± 18	256 ± 26	**460 ± 29**	835
Netherlands	328 ± 26	418 ± 21	419 ± 25	70 ± 20	307 ± 21	**488 ± 21**	*131*
Portugal	324 ± 27	407 ± 15	**483 ± 13**	-49 ± 13	265 ± 21	**462 ± 20**	606
Russia	382 ± 30	479 ± 24	**649 ± 14**	51 ± 21	419 ± 25	497 ± 23	732
Singapore	336 ± 29	386 ± 22	**566 ± 15**	-35 ± 14	274 ± 21	382 ± 19	*276*
Spain	433 ± 24	482 ± 17	**622 ± 14**	134 ± 24	358 ± 24	496 ± 15	759
UK	393 ± 28	456 ± 16	**538 ± 17**	138 ± 25	380 ± 22	471 ± 19	729
USA	263 ± 31	243 ± 28	**381 ± 33**	-119 ± 11	120 ± 25	238 ± 31	*-192*
Mean	342 ± 27	404 ± 22	**531 ± 17**	26 ± 19	295 ± 23	468 ± 21	*478*

Table 4: Comparison between CLUTR and other UED algorithms. Boldface denotes SOTA among UED algorithms, while italic in the Attention Agent colum means, CLUTR is comparable/outperforms the attention agent on that track. CLUTR outperforms PAIRED by a big margin with 18x bigger mean return on the entire F1 Dataset. CLUTR also outperforms Domain Randomization, PLR, and REPAIRED and only falls short to Robust PLR. Nonetheless, CLUTR shows competitive results compared to Robust PLR, showing comparable results in seven out of the 20 test tracks and outperforming in the Netherlands track. CLUTR also outperforms the non-UED SOTA on the full F1 dataset. CLUTR outperforms the Attention Agent on 9 out of the 20 tracks and shows comparableperformance in one. It must be noted, all the UED methods except CLUTR was trained for 5M timesteps where CLUTR was run for 2M timesteps.

(a) Mean Regret - Car Racing - with vs without flexible regret loss

(b) Returns on UED generated Car Racing tracks - with vs without flexible regret loss

Figure 12: Mean Regret and agent returns during training CLUTR (with flexible regret) vs CLUTR with standard PAIRED regret approximation.

CLUTR with a VAE trained on non-sorted data up to 5M timesteps. CLUTR-shuffled shows inferior performance and also signs of unlearning compared to CLUTR. Figure 16 shows detailed experiment results.

(a) Mean Regret - Car Racing

(b) Returns on UED generated Car Racing tracks

Figure 13: Mean Regret and agent returns during training CLUTR with standard PAIRED regret loss (i.e., without the flexible regret). CLUTR shows a smaller regret value(i.e., closely competing agent and antagonist), indicating a better UED curriculum.

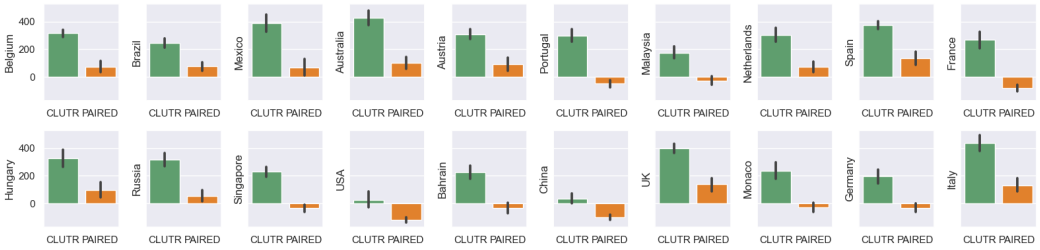


Figure 14: Zero-shot generalization of both PAIRED and CLUTR (with the standard regret loss) agents after 5M timesteps on the full F1 benchmark. CLUTR with the standard regret loss outperforms PAIRED on every track. For each track, we test the agents on 10 different episodes and the error bar denotes the standard error.

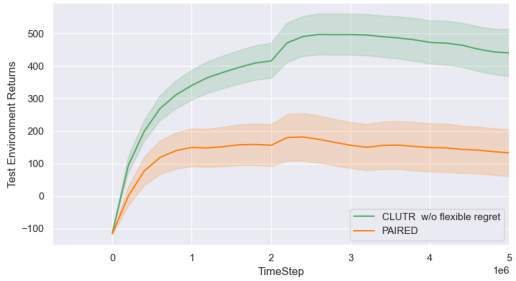
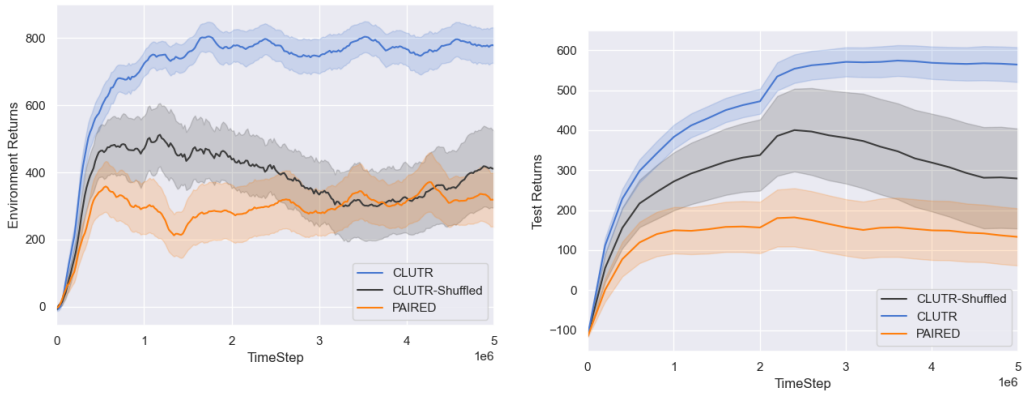


Figure 15: Test Returns on Selected Tracks (Vanilla, Singapore, Germany, and Italy) of CLUTR with standard PAIRED regret loss alongside PAIRED performance.



(a) During training CLUTR agent achieves higher returns while, CLUTR-shuffled agent shows lower returns. CLUTR-Shuffled agent's return is also less stable showing a decrease and increase.

(b) CLUTR achieves higher and more stable mean returns on the selected tracks. CLUTR-Shuffle shows signs of unlearning.

Figure 16: Analysis of sorting training data for VAE. Trained on shuffled data, CLUTR-Shuffled performs inferior compared to CLUTR and shows signs of unlearning.

C.4 Impact of Task Representation Learning

In this section, we discuss the impact of the learned task representation on performance. In Section 5.4, we showed that if we finetune the VAE decoder during curriculum learning, the overall performance drops significantly (Figure 6). To get a better understanding, in Figure 17, we plot how much the performance deviates as the VAE decoder changes during the training process. The curve in red shows the deviation of the decoder from its pretrained weights as it is fine-tuned during the training. We estimate the deviation as the L2 distance between the finetuned and the pretrained decoder weights. The green curve shows the performance drop from CLUTR (with standard loss). To estimate the performance drop, we periodically evaluate both CLUTR and CLUTR with Finetuned VAE, on the selected test tracks during training. From the figure, we observe that, as the decoder weights are finetuned, they become increasingly different from the initial pretrained weights. At the same time, the overall performance gap from CLUTR also increases. This suggests that the pretrained VAE weights are crucial for better performance.

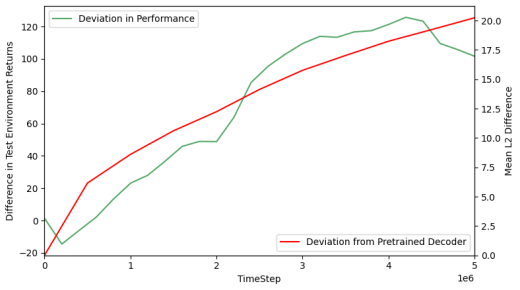


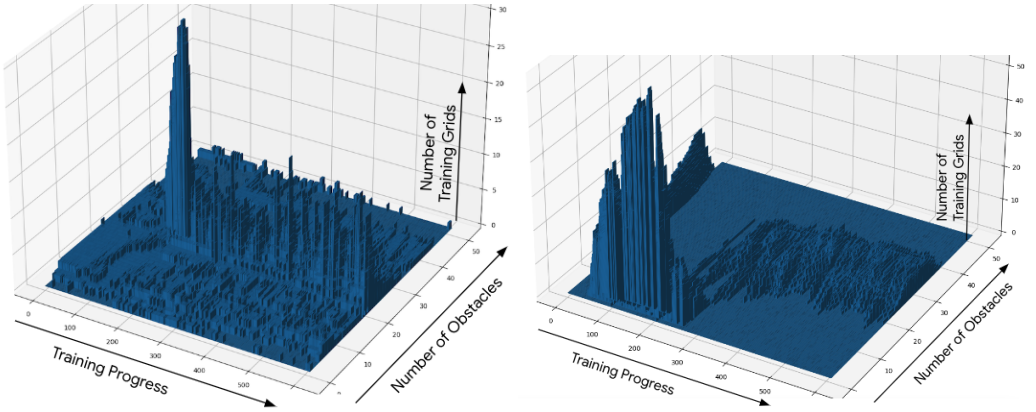
Figure 17: Impact of pretrained decoder weights on performance. The red curve plots the deviation of the decoder from its pretrained weights as it is finetuned. The green curve shows the performance drop from CLUTR with the standard loss. These curves suggest that pretrained weights are crucial for performance.

Furthermore, the quality of the learned representation depends on the quality of the data they are trained on. In section 5.5, we showed that a VAE trained on a non-sorted dataset significantly deteriorates the performance (Figure 6). This further suggests that the learned representation has a significant impact on performance. We also want to note that both of these variations (CLUTR with Finetuned VAE and the CLUTR with Shuffled VAE) perform much better than PAIRED, which suggests that, though CLUTR's performance depends on the representation, with a reasonable representation, it can still perform better than PAIRED.

D Detailed Results on MiniGrid

D.1 Curriculum Analysis

Figure 19 shows 3D Histograms showing the frequency of the generated grids against the total number of obstacles they contain. PAIRED starts with a high number of obstacles and then degenerates into grids with very few numbers of obstacles, eventually converging into a band of around 20 to 40 obstacles on average. CLUTR converges to a longer-tail band of 20 to 30 obstacles, much earlier in the training process. After the 'convergence', PAIRED rarely generates grids with fewer or more obstacles than the band it converges to. On the contrary, CLUTR still generates grids with few or many blocks, which might help to address unlearning or improve the agents on grids with more obstacles, respectively.



(a) CLUTR (b) PAIRED

Figure 18: 3D Histograms showing the frequency of the generated grids against the total number of blocks they contain. Both PAIRED and CLUTR converge to a similar band of grids. However, CLUTR converges much faster

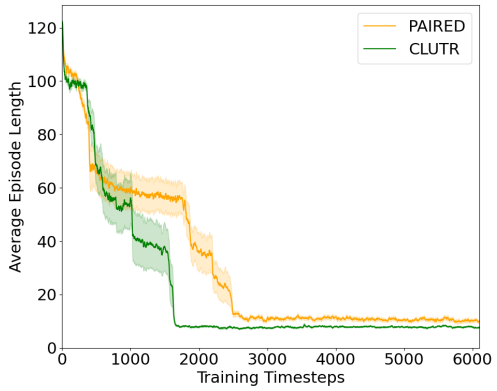
Figure 19a shows the average episode lengths of both CLUTR and PAIRED. The curves show both methods start with long episodes—indicating at the beginning, the agents do not solve the training grids consistently, and many of the episodes end due to timeout. As the agents learn, the episodes become shorter for both methods until they converge to a small value. However, CLUTR converges sooner than PAIRED.

We also compare the average solution length of the solved training grids. Both PAIRED and CLUTR show a similar pattern. However, PAIRED converges to a larger value than CLUTR. This might indicate that CLUTR is solving the environments more efficiently. This might also mean that CLUTR is solving some easier tasks (e.g., fewer obstacles, as we noticed from Figure 19) even after convergence lowering its average solved path length slightly.

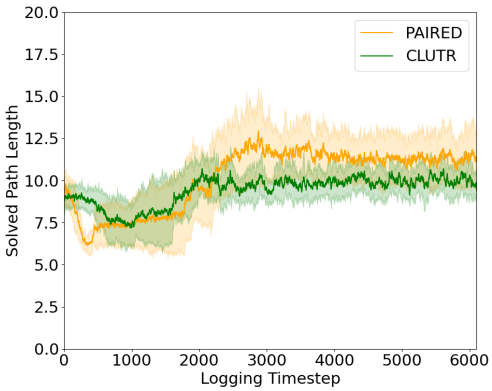
D.2 CLUTR curriculum vs. Domain Randomized curriculum on the latent space: Does CLUTR teacher degenerates into a randomized policy?

To answer whether CLUTR teacher actually learns something or degenerates into a randomized policy, we compare the curriculum generated by CLUTR with a random uniform (i.e., Domain Randomization) curriculum. We generate the DR curriculum by repeatedly sampling the trained VAE (the same VAE used by CLUTR) with a uniform random distribution. Figure 20 shows the comparison characterizing the grids by the number of obstacles they contain similarly as the previous section. As expected, we can see that the DR curriculum generates grids with obstacles ranging from 0 to 50. The histograms clearly visualize the significant differences in the curricula, implying the CLUTR teacher indeed learns a useful curriculum as suggested by the empirical result.

D.3 Analysis of the Latent Task Manifold

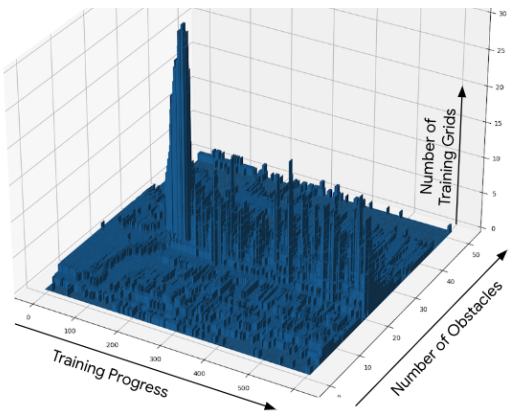


(a) Average length of the training episodes. CLUTR converges sooner than PAIRED to a shorter episode length.

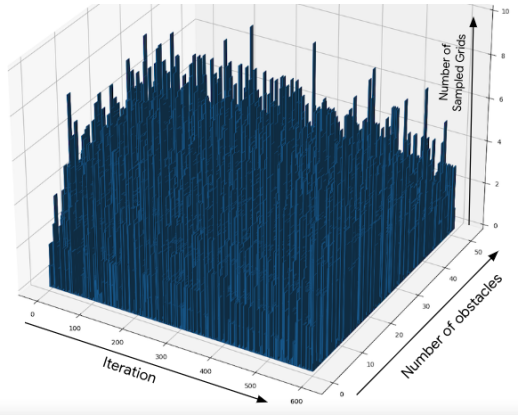


(b) Average solution length of the solved training tasks.

Figure 19: Comparison of CLUTR and PAIRED curriculum based on properties of the generated grids.



(a) CLUTR



(b) Domain Randomization of the Latent Space

Figure 20: 3D Histograms showing the frequency of the CLUTR generated grids against the total number of blocks they contain vs. randomly generated grids.

To grow a sense of the latent task manifold, we linearly interpolate in the latent space between an empty grid and a 15x15 version of the FourRoom grid (shown in Figure 21). Figure 22 visualizes the interpolation results. We first get the latent vectors of the empty grid and the target FourRoom task using the VAE encoder. We then linearly interpolate 23 equidistant points between them. At last, we reconstruct the grids from these vectors using our decoder. From Figure 22 we see that, as we interpolate in the latent space, the reconstructed grid incrementally adds more obstacles and the grids start to look more like the FourRoom target grid. We note that the reconstruction is not perfect. We also note that the increase in the number of obstacles is not uniform, e.g., the first 5 reconstructed grids are all empty grids, and more obstacles are added near the target point. Overall, this experiment provides an insight that the latent space holds a useful structure, which CLUTR teacher utilizes to generate the curriculum.

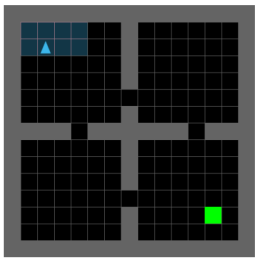


Figure 21: 15X15 FourRooms

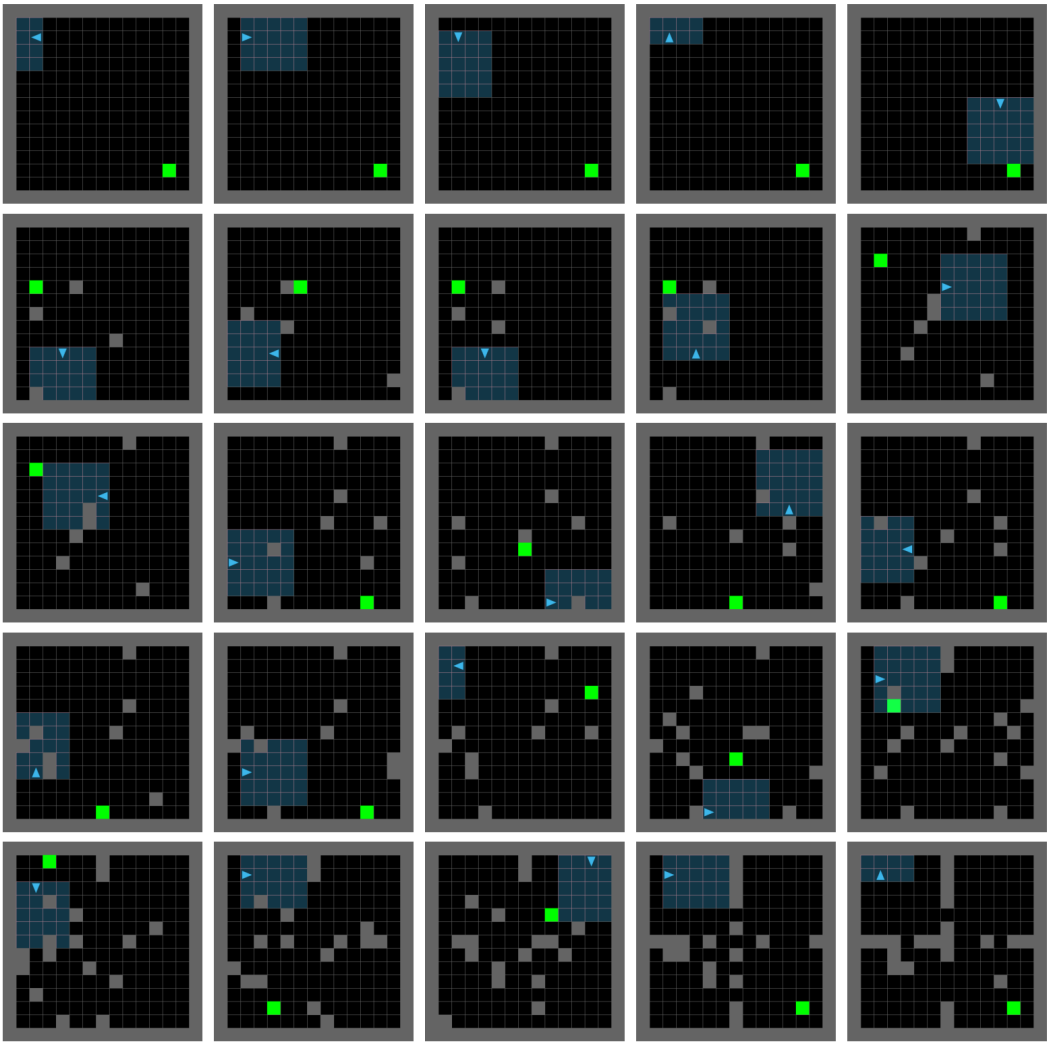


Figure 22: A linear interpolation between an empty grid and 15x15 version of the Four-Room grid (Figure 21) in the latent space. The grids are organized from top-left to bottom-right in row-major order.