RETHINKING LEARNING DYNAMICS IN RL USING ADVERSARIAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent years have seen tremendous progress in methods of reinforcement learning. However, most of these approaches have been trained in a straightforward fashion and are generally not robust to adversity, especially in the meta-RL setting. To the best of our knowledge, our work is the first to propose an adversarial training regime for Multi-Task Reinforcement Learning, which requires no manual intervention or domain knowledge of the environments. Our experiments on multiple environments in the Multi-Task Reinforcement learning domain demonstrate that the adversarial process leads to a better exploration of numerous solutions and a deeper understanding of the environment. We also adapt existing measures of causal attribution to draw insights from the skills learned, facilitating easier re-purposing of skills for adaptation to unseen environments and tasks.

1 INTRODUCTION

Recent years have seen tremendous progress in methods for reinforcement learning with the rise of "Deep Reinforcement Learning" (DRL; Mnih et al., 2015). In robotics, DRL holds the promise of automatically learning flexible behaviors end-to-end while dealing with multidimensional data, as mentioned in Arulkumaran et al. (2017). The level has risen to such heights that our algorithms are capable of learning policies that can defeat human professionals in many different games such as Chess, Go (e.g. by AlphaZero (Silver et al., 2017)), and many more complicated games such as Dota-2 (OpenAI, as presented in Berner et al. (2019)), robot control (Gu et al., 2016; Lillicrap et al., 2015; Mordatch et al., 2015), and meta-learning (Zoph & Le, 2016). Despite this recent progress, the predominant paradigm remains to train straightforward algorithms that are not robust to adversarial attacks, as shown in Zhou & Doyle (1998); Garcıa & Fernández (2015). In this work, we take a game-theoretic view to do adversarial training since doing so helps learn diverse skills with little supervision. As such, we draw intuition from the statement that "nothing makes you learn better than coevolving adversaries". Previous approaches try to create adversarial examples to make the models better suited for outliers (Pinto et al., 2017b; Chen et al., 2019), though these require manual intervention. Our work is novel in that we wish for the model to learn adversarially without manual intervention. Our proposed algorithm also learns the set of adversarial skills, leading to better exploration and more accessible domain adaptation to unseen tasks. This is achieved not by changing the tasks or rewards but by allowing the embedding network to learn multiple skill distributions without domain knowledge of the environment or the task at hand.

1.1 PROBLEM STATEMENT

In this work, we consider the Multi-Task Reinforcement Learning or the Meta-RL setting. The goal here is to teach a given agent to perform a set of tasks ($\{\tau_1, \tau_2, ..\tau_n\}$), where in most cases, all tasks garner equal rewards for the goal state. The problem is interesting since manual intervention is not scalable here, and exploration is more crucial than the standard reinforcement setting. For instance, it is a sub-optimal solution when the agent learns to solve only one of the tasks, say τ_1, completely ignoring the other tasks $\{\tau_2, ..., \tau_n\}$. If all the tasks give proportional rewards, the agent would have solved the environment without the incentive to explore the other tasks. To alleviate this, we propose the adversarial training regime that brings robustness and an added exploration factor to our policy.

1.2 Contributions

In this section, we present the main contributions of the paper:

A novel adversarial training regime: The main contribution of our work is an adversarial training regime for on-policy optimization using concepts from game theory literature, enabling us to learn distinct and robust latents for each task. To the best of our knowledge, our work is the first to propose an adversarial training regime for Multi-Task Reinforcement Learning, which requires no manual intervention or domain knowledge of the environments. (see Section 3)

Theoretical guarantees: We also provide theoretical guarantees that aid optimization in the adversarial regime (see Appendix C). Following the results from the above proof will help readers avoid the generic limitations of an adversarial training regime, as highlighted in Appendix E. Furthermore, we also show that the skills learned are mutually exclusive to the task. This property is helpful to re-purpose the skills for different and unseen tasks, as discussed in the upcoming sections.

A Causal Insight into RL: We propose an efficient framework to approximate the average causal effect of each embedding on the reward obtained for a given task in the on-policy domain. This facilitates a deeper understanding of the latent variables and helps re-purpose them for other unseen tasks (see Section 4). With our proposed causal attribution framework, we can also show that the model can learn multiple solutions with the minimum number of distinct skills necessary for the given set of tasks. (see Appendix G)

Experimental Results: We demonstrate that our method outperforms our baseline in several simulated environments such as PointMass, 2D navigation, and robotic manipulation tasks from Meta-World. (see Section 5)

Results on Domain Adaptation: We also show that the model trained using the adversarial training regime leads to better generalization and more accessible domain adaptation to unseen and complex tasks, even in sparse rewards (see Section 5.3). This result is per the added exploration factor the agent benefits from when trained in the adversarial regime. This exploration factor has been studied under the context of embedding efficiency of latents. (see Appendix 2)

2 Preliminaries

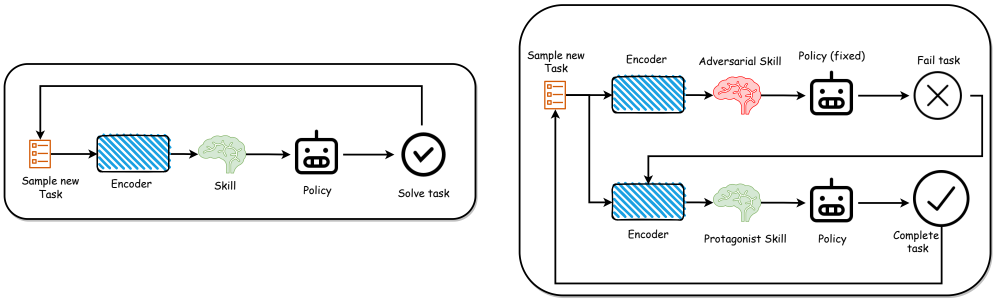
Before we delve into the details of our adversarial training regime-based algorithm, we first outline our terminology, standard reinforcement learning setting, and two-player adversarial games from which our paper is inspired.

2.1 Standard reinforcement learning in MDPs

We study reinforcement learning in Markov Decision Processes (MDPs). We denote $s \in \mathbb{R}^S$ as the continuous state of the agent; $a \in \mathbb{R}^A$ as the action vector and $p(s_{t+1}|s_t, a_t)$ as the probability of transitioning from state s_t to s_{t+1} when executing action a_t. Actions are drawn from a policy distribution $\pi_\theta(a|s)$, with parameters θ. This work uses a Gaussian distribution whose mean and diagonal covariance are parameterized via a neural network. At every step, the agent receives a scalar reward $r(s_t, a_t)$ and the goal is to maximize the sum of discounted rewards: $\mathbb{E}_{\tau_\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. We use TE-PPO (Hausman et al., 2018) as our baseline since it also uses latent skills for learning policy and serves as the perfect baseline to showcase the effect of an adversarial training regime in the Meta-RL setting.

2.2 Two-player adversarial games

The adversarial setting we propose can be expressed as a two-player γ-discounted non-zero-sum Markov Game. Instead of learning the policy directly in an adversarial manner, we choose to learn the skills used by the policy in an adversarial manner. Under this setting, the MDP is similar to the previous notion, with an added variable from the adversary – a skill that reduces the average discounted return on a given task. Player 1 (Encoder + policy) will try to act as a protagonist and solve the given task. In contrast, Player 2 (Encoder alone) will serve as an adversary and learn skills that best fool the protagonist for the given task. While Player 2 acts, the policy network is frozen, and



(a) Standard Learning Regime

(b) Adversarial Learning Regime

Figure 1: Figure 1(a) and Figure 1(b) depicts the straightforward (TE-PPO) and adversarial learning regime (ATE-PPO) respectively. Note that the same task is used in both learning cycles in the Adversarial Training Regime. Furthermore, the top sequence in Figure 1(b) tries to minimize reward/maximize regret when the policy is fixed (learning adversarial skills). In contrast, the bottom sequence of the same algorithm tries to correct this adversity by learning both the encoder and the policy.

only the embedding network is learned. We call our algorithm **Adversarial TE-PPO** (or ATE-PPO) and describe our model in more detail in Section 3. A brief overview of our algorithm is presented in Algorithm 1. As discussed in the subsequent section, our end-objective function works in the minimax domain and differentiates our work from TE-PPO.

Figure 1 depicts the high-level working of TE-PPO and ATE-PPO, respectively. We encourage readers to read the respective paper for further details about the TE-PPO algorithm.

3 Adversarial Training Regime in RL

Our approach, similar to the setup of TE-PPO, and comprised of three different networks which are trained in an adversarial training regime: an Encoder network ($\mathcal{E}$), a Policy network (π), and an Inference network ($\mathbb{I}$). The Encoder tries to learn a skill embedding – something unique for each task at hand, and the Policy network uses this embedding to select actions maximizing its reward. The adversarial modeling framework is most straightforward when the models are multilayer perceptrons. Instead of training two different sets of agents/policies in an adversarial fashion, we do so at the skill/encoder level. We will first learn the encoder parameters that provide the "Worst Faithful Embedding" - skills that reduce the discounted rewards for the given policy and minimize mutual information between the skills and one-hot encoding of tasks. By doing so, any spurious correlations between skills would disappear since we are learning for the worst case. We then learn the agent initialized at the given "Worst Faithful Embedding" to learn policies immune to adversarial perturbations.

Our mathematical formulation of the above intuition would follow along, as shown. To learn the skill embedding z, we represent a mapping between the one-hot encoded task embedding τ to a continuous skill space z as $z = \mathcal{E}(\tau; \theta_e)$; where $\mathcal{E}$ is a differentiable function represented by a multilayer perceptron with parameters θ_e and z is the skill required for achieving task τ. To take advantage of these skill embeddings, we define a second multilayer perceptron $\pi(z; \phi_\pi)$ that helps the agent learn the actions to take and maximize its reward. Similar to Hausman et al. (2018), we also allow the encoder to learn in this step to facilitate learning optimal skills for the given task. To achieve this training regime, our model plays a two-player minimax game where the agent will try to learn a good policy and embedding with a protagonist loss function $\mathcal{L}_{\text{pro}}$:

$$\mathcal{L}_{\text{pro}} = -\mathbb{E}_{\pi, p_0, \tau \in \mathcal{T}} \left[Q_\pi(s, a; \tau) \right] \tag{1}$$

However, the adversary perturbs the embedding slightly to fail at the task, which you cannot detect by computing mutual information between task and embedding. The adversary will work with a fixed policy and an adversarial loss function $\mathcal{L}_{\text{adv}}$:

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{\pi,p_0,t\in\mathcal{T}}\left[Q_\pi(s,a;\tau)\right] - \alpha\mathbb{E}_{\tau\in\mathcal{T}}\left[\mathcal{H}(z)+\mathcal{H}(\tau)-\mathcal{H}(\tau|z)\right] \tag{2}$$

Hence, you are forced to learn a good policy and embedding which work despite these perturbations. Here $p_0(s_0)$ is the initial state distribution, α is a weighing term – a trade-off between deceiving the agent network and minimizing the entropy terms denoted by $\mathcal{H}(.)$. Furthermore, the Q-function $Q_\pi(s,a,\tau)$ is defined as:

$$Q_\pi(s,a,\tau) = \sum_{i=0}^{\infty}\gamma^i(r_\tau(s_i,a_i) + \alpha_0\mathcal{H}[\pi(a_i|s_i,\tau)]) \tag{3}$$

where action a_i is drawn from the distribution $\pi(.|s,\tau)$, and the new state is $s_{i+1}\sim p(s_{i+1}|a_i,s_i)$. Note that α_0 is a constant and is treated as a hyperparameter. The first half of the equation defined by $\sum_{i=0}^{\infty}\gamma^i r_\tau(s_i,a_i)$ is the discounted expected returns and is common to our generic reinforcement learning algorithms. The latter is an entropy regularization term defined by $\sum_{i=0}^{\infty}\gamma^i\mathcal{H}[\pi(a_i|s_i,\tau)]$ which is conventionally applied to many policy gradient schemes, with the critical difference that it takes into account not only the entropy term of the current but also future actions. This approach has been used before by Hausman et al. (2018), but with a different objective function and trained in a straightforward reinforcement learning fashion, without any adversarial training dynamics.

To apply this entropy regularization to our setting of latent variables (skill embeddings), some extra mathematical rigor is required. Borrowing from the toolkit of variational inference (Agakov, 2004), we can construct a lower bound on the entropy term from Equation 3 as (see Appendix B) such that:

$$\mathbb{E}_{\pi,p_0,\tau\in\mathcal{T}}\left[Q_\pi(s,a,\tau)\right] = \mathbb{E}_{\pi(a,z|s,\tau)}\left[Q_\pi^\varphi(s,a;z,\tau)\right] + \alpha_1\mathbb{E}_{\tau\in\mathcal{T}}\mathcal{H}[p(z|\tau)]$$

where,

$$Q_\pi^\varphi(s,a;z,\tau) = \sum_{i=0}^{\infty}\gamma^i\widehat{r}(s_i,a_i,z,\tau)$$

Here, $s_{i+1}\sim p(s_{i+1}|a_i,s_i)$, and $\widehat{r}(s_i,a_i,z,\tau) = \left[r_t(s_i,a_i) + \alpha_2\log[\mathbb{I}(z|a_i,s_i^H)] + \alpha_3\mathcal{H}[\pi(a|s,z)]\right]$. Note that this is where the inference network $\mathbb{I}$ comes into play. Furthermore, H is the history of states saved in the buffer used for the inference. From the above equation, we simplify our objective function as follows and set α_1 as α, where we denote Mutual Information by $\mathcal{I}$, and Jensen Shannon Divergence by $\mathcal{JSD}$:

$$\begin{aligned}\mathcal{L}_{\text{adv}} &= \left[Q_\pi^\varphi(s,a;z,\tau)\right] - \alpha\mathbb{E}_{\tau\in\mathcal{T}}\left[\mathcal{H}(z)-\mathcal{H}(z|\tau)+\mathcal{H}(\tau)-\mathcal{H}(\tau|z)\right]\\ &= \left[Q_\pi^\varphi(s,a;z,\tau)\right] - \alpha\mathbb{E}_{t\in\mathcal{T}}\left[\mathcal{I}(z;\tau)+\mathcal{I}(t;\tau)\right]\\ &= \left[Q_\pi^\varphi(s,a;z,\tau)\right] - \alpha\mathbb{E}_{\tau\in\mathcal{T}}\left[\mathcal{JSD}(z,\tau)\right]\end{aligned} \tag{4}$$

Although, we use the objective presented in Equation 4 to show theoretical guarantees (see Appendix C), we simplify the equation further, making the implementation tractable (see Appendix D). In our implementation, we use the modified objective presented in Appendix D in an interleaved fashion. This is similar to adversarial networks in the on-policy setting. Furthermore, we would also like to extend this training regime to obtain a data-efficient, off-policy algorithm that could be applied to a natural robotic system in the future. (see Appendix A)

4 Causal Perspective into RL

Using latent embeddings as skills can aid the study of the causality of the skill upon the task at hand. In this work, we propose a simple framework to compute the Average Causal Attribution of the skills on the task: the average difference between potential outcomes under different treatments. The Average Causal Effect (ACE) of a variable x on another random variable y is formally defined as:

$$ACE^y_{do(x_i=\alpha)} = \mathbb{E}[y|do(x_i=\alpha)] - baseline_{x_i} \tag{5}$$

The ACE requires the computation of two quantities: the interventional expectation ($\mathbb{E}[y|do(x_i=\alpha)]$) and the baseline ($baseline_{x_i}$). Here, do corresponds to the do-operation (Pearl, 2009) of fixing a variable x_i to a specific value α.

Algorithm 1: Our proposed Adversarial Training Regime for Reinforcement Learning

```
Input: Data sampling mechanism $\mathbb{T}$, Data $\mathcal{D}$, number of episodes $N$, Encoder Networks $\mathcal{E}$,
       Policy Network $\pi$, Number of Adversarial steps $\mathcal{A}$, and Number of Protagonist steps $\mathcal{P}$.
$n \leftarrow 0$;
$\tau_i \leftarrow \mathsf{OneHot}(i)$                          ▷ Create one-hot vectors of size k
$\tau \leftarrow \{\tau_1 \quad \tau_2 \ldots \quad \tau_k\}$                     ▷ Create Task Embedding
while $n < N$ do
    $n \leftarrow n + 1$;
    $\{\mathcal{T}_1 \quad \mathcal{T}_2 \ldots \quad \mathcal{T}_k\} \leftarrow \mathbb{T}(\mathcal{D})$           ▷ Sample k tasks from data distribution.
    $a \leftarrow 0$;
    $p \leftarrow 0$;
    while $a < \mathcal{A}$ do
        $a \leftarrow a + 1$;
        $\min_{\mathcal{E}} \mathcal{L}_{\text{adv}}$                  ▷ Learn adversarial skills keeping policy fixed
    end
    while $p < \mathcal{P}$ do
        $p \leftarrow p + 1$;
        $\min_{\mathcal{E},\pi} \mathcal{L}_{\text{pro}}$          ▷ Learn protagonist skills learning both policy and encoder
    end
end
```

Interventional Expectation The interventional expectation is a function of x_i as all other variables are marginalized. By definition:

$$\mathbb{E}[y|do(x_i = \alpha)] = \int_y yp(y|do(x_i = \alpha))dy \tag{6}$$

Naively, evaluating the interventional expectation would involve sampling all other input features from the empirical distribution, keeping feature $X_i = \alpha$, and then averaging the output values. However, this computation assumes that the input features do not cause each other and would be time-consuming to run for the entire training data. Furthermore, we also assume that the features are d-separated: given an intervention on a particular variable, the probability distribution of all other neurons does not change, i.e. $p(x_j|do(x_i = \alpha)) = p(x_j)$, if $i \neq j$. Since the latent learned by our models are time-agnostic, our assumption is justifiable. To make the computation feasible, we make use of first-order Taylor expansion of the causal mechanism $f(x|do(x_i = \alpha))$ around the mean vector $\mu = [\mu_1, \mu_2, \cdots, \mu_k]^T$ is given by:

$$f(x|do(x_i = \alpha)) \approx f(\mu|do(x_i = \alpha)) + \nabla f(\mu|do(x_i = \alpha))^\top (x - \mu|do(x_i = \alpha)) \tag{7}$$

Taking expectations on both sides (marginalizing over all other input neurons), we arrive at the Interventional expectation:

$$\mathbb{E}[f(x|do(x_i = \alpha))] \approx f(\mu|do(x_i = \alpha)), \tag{8}$$

Where $f(.)$ is the average reward for a given task, x is the input vector or the latent embedding, and μ is the mean of input vectors when $x_i = \alpha$. Note that the first-order term vanishes since $\mathbb{E}(x|x_i = \alpha) = \mu$. This is extremely important since computing the gradient through a step in a reinforcement learning environment is impossible since the transition function is generally non-differentiable. To make it differentiable, additional approximations and workarounds would be needed to achieve the same. Hence, we do not extend the same Taylor approximation to the second-order or higher for the same reason. Such approximations of deep non-linear neural networks via Taylor's expansion have been explored in the context of causality (Chattopadhyay et al., 2019). However, their overall goal was different from the reinforcement learning domain.

Baseline An ideal baseline would be any point along the decision boundary where predictions are neutral. However, as showed by the work performed by Kindermans et al. (2019), when the reference baseline is fixed to a specific value, attribution methods are not affine invariant. Instead, we define baseline as:

$$baseline_{x_i} = \mathbb{E}_{x_i}[\mathbb{E}_y[y|do(x_i = \alpha)]] \tag{9}$$

Rationally, the baseline is defined as $\mathbb{E}_y[y|do(x_i = \widehat{x_i})]$, which includes all values of the input feature. If the expected value y is constant for all possible intervention values α, the baseline would also be the same constant and result in the causal attribution $ACE^y_{do(x_i=\alpha)} = 0$. Furthermore, the L1-norm of the causal attribution – $\mathbb{E}_{x_i}[|\mathbb{E}_y[y|do(x_i = \alpha)]|]$ could serve as an metric of "**Average Importance**" of a given attribute. This importance metric is then normalized for easier interpretation. Note that the proposed framework is model-agnostic and can be adapted to similar models which work with latents (TE-PPO and ATE-PPO in this paper). This paper's causal attributions have been computed over the agent trained on ATE-PPO. Although the approximation is very coarse, we can still obtain various insights from the proposed framework. (see Appendix G)

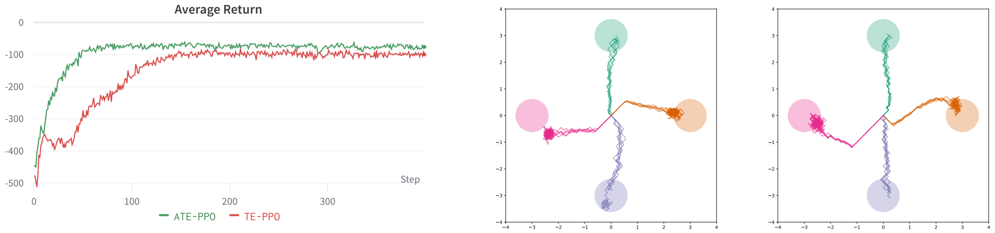
5 Experimental Results and Discussions

We evaluate our approach in two domains in simulation: simple environments such as PointMass task, 2-D navigation task, and a set of challenging robot manipulation tasks from Meta-World. We consider the Gaussian embedding space for all experiments for both algorithms (ATE-PPO and TE-PPO).

5.1 Learning versatile skills

We highlight this property of learning versatile skills from our experiments on relatively more straightforward environments such as the PointMass environment and 2-D Navigation environment.

PointMass Environment Similar to Haarnoja et al. (2017), we present a didactic example of multi-goal PointMass tasks demonstrating the variability of solutions that our method can discover. In this experiment, we consider a case where four goals are located around the initial location, and each is equally important to the agent. This leads to a situation where multiple optimal policies exist for a single task. In addition, this task is challenging due to the sparsity of the rewards – as soon as one solution is discovered, it becomes exceedingly difficult to keep exploring other goals. Due to these challenges, most existing DRL approaches would be content with finding a single solution. Furthermore, even if a standard policy gradient approach discovered multiple goals, it would have no incentive to represent various solutions. However, with the added goal to maximize $\mathcal{H}(z)$, our ATE-PPO algorithm should lead to better exploration and learning of different skills. Our proposed approach outperforms the baseline, as depicted in Table 1. Figure 2 expresses more insights into the learning trajectory and the final agent behavior.



(a) Learning Curve (b) TE-PPO (c) ATE-PPO

Figure 2: Figure 2(a) depicts the Average return on PointMass Environment using TE-PPO and ATE-PPO algorithms. The above illustration shows that the adversarial training regime boosts performance in the PointMass environment. Figure 2(b) and Figure 2(c) show the resulting path taken by the agent trained on TE-PPO and ATE-PPO, respectively (10 random agents trying to solve each task, where the four colored circles are the goals).

To better understand the skill learned by our model, we use the causal framework proposed in Section 4 to compute the average causal attribution of each feature of the skill vector. The average importance of each component is shown in Figure 3. This result is interesting because the feature z_2 consistently does not contribute to the agent's behavior. Although the latent space is ample, and the model has sufficient capacity, our policy network can learn only the minimum number of skills required to solve the task.

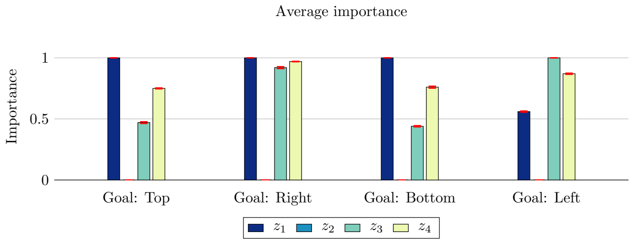
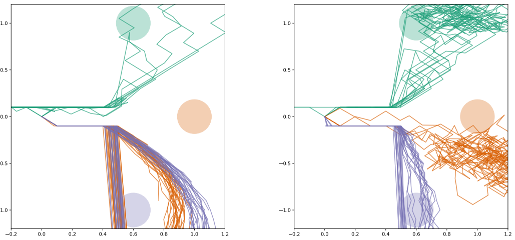


Figure 3: The normalized importance of each latent feature on the PointMass tasks across five different seeds.



(a) TE-PPO

(b) ATE-PPO

Figure 4: Figure 4(a) and Figure 4(b) show the resulting trajectories by the agent trained on TE-PPO and ATE-PPO respectively. Note that the three circles represent the goals for different tasks. The above illustration shows that the adversarial training regime boosts performance in the 2-D Navigation environment. Furthermore, the skills learned can create diversity and exploration of the agent in the future instead of the present.

2-D Navigation Environment Furthermore, we also experiment on the 2-D navigation task (Finn et al., 2017) with a slight modification. At each time step, the 2D agent takes action (its velocity, clipped in $[-0.1, 0.1]$) and receives a penalty equal to its L2 distance to the goal position (i.e., the reward is "$-$distance"). However, we restrict the agent's movement to a small corridor at the beginning, so the vertical state space is clipped between $[-0.2, 0.2]$. In this paper, we sample three tasks to solve in the current experimental setup. The agent starts from $(0, 0)$, and the goals of the tasks are to reach respective points in the Cartesian plane: $(1, 0)$, $(0.6, 1)$, and $(0.6, -1)$ denoted by various colors in Figure 4. We find that the ATE-PPO model can successfully learn the task and achieve better performance than the TE-PPO algorithm, as depicted in Table 1. We notice that our ATE-PPO model successfully learns skills that are diverse in the future and not the present. Figure 4 shows the resulting trajectories by the agent trained on TE-PPO and ATE-PPO, respectively. Furthermore, in more detail, we discuss the ACE plots in Appendix G.2. Our proposed approach outperforms the baseline, as depicted in Table 1.

5.2 Learning optimal representations

Next, we evaluate whether we can learn better representations with the help of the adversarial training regime. We create a pool of tasks from Meta-World with an overlapping skill set for most tasks in this problem. In our experiment, we set this overlapping skill as the act of "pushing". With this constraint, we create an environment MT5 by selecting five tasks for training: Push, Open window, Close window, Open drawer, and Close drawer. The multi-task evaluation tests the ability to learn multiple tasks simultaneously without accounting for generalization to new tasks. This task helps evaluate the efficiency of the skills learned by the model. ATE-PPO significantly improves TE-PPO's performance without significant hyperparameter tuning and using common network architecture across both experiments. TE-PPO achieves an average success rate of 0.2 ± 0.01, while our ATE-PPO achieves an average success rate of 0.41 ± 0.08. We summarize more detailed results of the causal analysis of latent in Appendix G.3.

5.3 Generalizing to unseen environments

Next, we evaluate whether our model can adapt to a new unseen task with minimum adaptation steps. We use our pre-trained model from the previous section for this task, which was trained on the MT5 environment. We selected a diverse set of tasks for this experiment with varying degrees of domain shift:

- **Push Wall:** The goal of this task is similar to the initial Push task from MT5, but we now also have to bypass a wall to reach the goal.
- **Coffee Button:** The goal of this task is to press a button on the coffee machine. We randomize the position of the coffee machine to avoid overfitting.
- **Push Back:** The goal of this task is to push the same object backward, similar to the Push task from MT5. We also randomize puck positions to avoid overfitting.
- **Faucet Open:** The goal of the task is to turn on the faucet. Specifically, we would like to rotate the faucet counter-clockwise and randomize faucet positions to avoid overfitting.

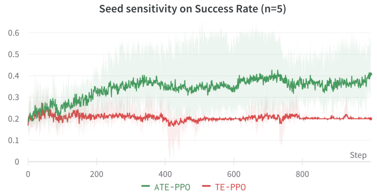


Figure 5: Average success rate on MT5 Environment using TE-PPO and ATE-PPO algorithm. The above illustration shows that the adversarial training regime does offer a boost in performance in the MT5 environment.

We run the agent for all the above domain adaptation tasks for ten epochs. Although the domain-adapted code is not sufficiently trained to solve the task perfectly, our pre-trained ATE-PPO model outperforms the TE-PPO across most target tasks (if not, competitive), as shown in Table 1. Furthermore, we run both models in the standard reinforcement learning setting without any adversarial training regime for better comparison.

6 Related Works

This section describes prior studies broadly related to the adversarial training regime of skills. Below, we divide prior research under two prominent subheadings: (i) Adversarial training regime and (ii) Multi-Task reinforcement learning.

Table 1: Average return on simulated environments over five seeds.

Environment	TE-PPO	ATE-PPO
2D Navigation	-170.03 ± 47.98	**-102.39** ± 44.44
Push Wall	2.54 ± 0.02	**2.69** ± 0.18
Coffee Button	37.21 ± 0.82	**80.91** ± 2.80
Push Back	0.85 ± 0.01	**0.89** ± 0.05
Faucet Open	474.41 ± 269.73	**584.29** ± 237.43

6.1 Adversarial Training Regime

In reinforcement learning, learned policies should be robust to uncertainty and parameter variation to ensure predictable behavior. Furthermore, learning policies should employ safe and effective exploration with improved sample efficiency to reduce the risk of costly failure. These issues have long been recognized and studied in reinforcement learning (Zhou & Doyle, 1998; Garcıa & Fernández, 2015). However, these issues are exacerbated in deep RL using neural networks, which, while more expressible and flexible, often require more data to train and produce potentially unstable policies. In Pinto et al. (2017a), two supervised agents were trained, with one acting as an adversary for self-supervised learning, which showed an improvement in robot grasping. Other adversarial multiplayer approaches have been proposed, including Heinrich & Silver (2016). However, these approaches require the training of two agents. Inspired by these, Heinrich & Silver (2016) proposed an approach to learning two policies for a single agent, one as a protagonist and the other as an adversary. However, this requires hard-coding parts of the agent where an adversarial force is applied to fool the model. Although these approaches lead to a more robust model, our goal in this paper is slightly different.

Instead, our goal is to propose an adversarial training regime without significant hard coding. We perform the adversarial training regime at the skill level, allowing better exploration. Since the same policy network is used in both cases, the policy understands the effect of wrong actions, increasing understanding of the environment due to the increased exploration of the state space.

6.2 Multi-Task reinforcement learning

Learning multiple tasks at once in reinforcement learning is quite common with the intuition that more tasks would suggest higher diversity and hence better generalization in downstream tasks. This property is best showcased in the work (Eysenbach et al., 2018), where they learn diverse skills without any reward function. Furthermore, sequential learning and the need to retain previously known skills have always been a focus (Rusu et al., 2016; Kirkpatrick et al., 2017). In the space of multi-task reinforcement learning with neural networks, Teh et al. (2017) proposed a framework that allows sharing knowledge across tasks via a task agnostic prior. Similarly, Cabi et al. (2017) uses off-policy learning to learn about many different tasks while following a primary task. Other approaches (Devin et al., 2017; Denil et al., 2017) propose architectures that can be reconfigured easily to solve various tasks. Subsequently, Finn et al. (2017) uses meta-learning to acquire skills that can be fine-tuned effectively.

The use of latent variables and entropy constraints to induce diverse skills has been considered before (Daniel et al., 2012; End et al., 2017) albeit in a different framework and without using neural network function approximators. Further, Konidaris & Barto (2007) used the options framework to learn transferable options using the so-called agent space. Inspired by these ideas, Hausman et al. (2018) introduces a skill embedding learning method that uses deep reinforcement learning techniques and can concisely represent and reuse skills. Their works draws on a connection between entropy-regularized reinforcement learning and variational inference literature (Todorov, 2008; Toussaint, 2009; Neumann et al., 2011; Levine & Koltun, 2013; Rawlik et al., 2013; Fox et al., 2015). The notion of latent variables in policies has been previously explored by works such as controllers (Heess et al., 2016), and options (Bacon et al., 2017). The auxiliary variable perspective introduces an informative-theoretic regularizer that helps the inference model produce more versatile behaviors. Learning these versatile skills has been previously explored by Haarnoja et al. (2017), which learns an energy-based, maximum entropy policy via a Q-learning algorithm, and Schulman et al. (2017) which uses an entropy regularized reinforcement learning policy. Hausman et al. (2018) uses a similar entropy-regularized reinforcement learning and latent variables but differs in the algorithmic framework.

7 Conclusion

We present an adversarial training regime that learns manipulation skills continuously parameterized in a skill embedding space. Our algorithm takes advantage of these skills and creates an adversary at the skill level instead of the policy level. The skills are learned by taking advantage of the increased explorability of known variables and exploiting a connection between reinforcement learning, variational inference, adversarial networks, and minimax algorithm from game theory. We derived an adversarial training regime for reinforcement learning and demonstrated its robustness in multiple environments. Primarily, we showed that our model could learn skills that diverge in the future rather than the present. We also introduce a methodology for computing ACE, which would help re-purposing skills and using them efficiently in o.o.d environments. Our experiments with the help of ACE also indicate that even in the presence of enough capacity, our model is capable of discovering multiple solutions and learning the minimum number of distinct skills and latents necessary to solve the given tasks. Furthermore, the learned skills are more adaptable and make for better cross-domain adaptation than the traditional reinforcement learning approach. We believe the proposed framework is more ubiquitous than just reinforcement learning and could be adapted to other domains such as supervised learning, representation learning, and many others.

Reproducibility Statement

In this paper, we work with three different datasets, which are all open-sourced. Furthermore, we experiment with two different models - TE-PPO and ATE-PPO. The TE-PPO model was run after reproducing from their open-source code[1]. Additional details about setting up these models, and the hyperparameters are available in Appendix H. Our source code is made available for additional reference as a supplementary file for your reference.

References

David Barber Felix Agakov. The im algorithm: a variational approach to information maximization. *Advances in neural information processing systems*, 16(320):201, 2004.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw 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.

Serkan Cabi, Sergio Gómez Colmenarejo, Matthew W Hoffman, Misha Denil, Ziyu Wang, and Nando Freitas. The intentional unintentional agent: Learning to solve many continuous control tasks simultaneously. In *Conference on Robot Learning*, pp. 207–216. PMLR, 2017.

Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N Balasubramanian. Neural network attributions: A causal perspective. In *International Conference on Machine Learning*, pp. 981–990. PMLR, 2019.

Tong Chen, Jiqiang Liu, Yingxiao Xiang, Wenjia Niu, Endong Tong, and Zhen Han. Adversarial attack and defense in reinforcement learning-from ai security view. *Cybersecurity*, 2(1):1–22, 2019.

Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pp. 273–281. PMLR, 2012.

Misha Denil, Sergio Gómez Colmenarejo, Serkan Cabi, David Saxton, and Nando de Freitas. Programmable agents. *arXiv preprint arXiv:1706.06383*, 2017.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2169–2176. IEEE, 2017.

Felix End, Riad Akrour, Jan Peters, and Gerhard Neumann. Layered direct policy search for learning hierarchical skills. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6442–6448. IEEE, 2017.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[1] https://github.com/rlworkgroup/garage

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pp. 2829–2838. PMLR, 2016.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pp. 1352–1361. PMLR, 2017.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 267–280. Springer, 2019.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

George Dimitri Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pp. 895–900, 2007.

Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. *Advances in neural information processing systems*, 26:207–215, 2013.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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.

Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, and Emanuel V Todorov. Interactive control of diverse complex characters with neural networks. *Advances in Neural Information Processing Systems*, 28:3132–3140, 2015.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G Bellemare. Safe and efficient off-policy reinforcement learning. *arXiv preprint arXiv:1606.02647*, 2016.

Gerhard Neumann et al. Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pp. 817–824, 2011.

Judea Pearl. *Causality*. Cambridge university press, 2009.

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, 2017a.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pp. 2817–2826. PMLR, 2017b.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Twenty-third international joint conference on artificial intelligence*, 2013.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

Emanuel Todorov. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pp. 4286–4292. IEEE, 2008.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1049–1056, 2009.

Kemin Zhou and John Comstock Doyle. *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A Extending to Off-Policy Setting

In this section, estimate the discounted sums from Equation 4 from previously gathered data by learning a Q-value function, yielding an off-policy algorithm. Similar to Hausman et al. (2018), we assume the availability of a replay buffer $\mathcal{B}$ (containing full trajectory execution traces including states, actions, task id, and reward) that is inherently filled during training. In conjunction with these trajectory traces, we also store the probabilities of each selected action and denote them with the behavior policy probability $b(a|z, s, \tau)$ and the behavior probabilities of the embedding $b(z|\tau)$. Given this replay data, we formulate the off-policy perspective of our algorithm. We start with the notion of a *lower-bound Q-function* that depends on both states s and a and is conditioned on both the embedding z and the task id τ. To learn a parametric representation of Q^{φ}_{π}, we make use of the Retrace algorithm (Munos et al., 2016), which quickly allows us to propagate entropy augmented rewards across multiple time steps while minimizing the bias of the algorithm relying on the parametric Q-function. Formally, we fit Q^{φ}_{π} by minimizing the squared loss:

$$\begin{aligned} &\min_{\varphi} \mathbb{E}_{\mathcal{B}}\left[(Q^{\pi}_{\varphi}(s_i, a_i; z, \tau) - Q^{\text{ret}})^2\right], \text{ where} \\ &Q^{\text{ret}} = \sum_{j=i}^{\infty}\left(\gamma^{j-i}\prod_{k=i}^{j} c_k\right) r(s_j, a_j, z, \tau) \end{aligned} \tag{10}$$

wherein,

$$\begin{aligned} r(s_j, a_j, z, \tau) &= \widehat{r}(s_j, a_j, z, \tau) + \mathbb{E}_{\pi(a|z,s,\tau)}\left[Q^{\pi}_{\varphi'}(s_i, .; z, \tau) - Q^{\pi}_{\varphi'}(s_j, a_j; z, \tau)\right] \\ c_k &= \min\left(1, \frac{\pi(a_k|z, s_k, \tau)p(z|\tau)}{b(a_k|z, s_k, \tau)b(z|\tau)}\right) \end{aligned}$$

We compute the terms contained in $\widehat{r}$ by using r_τ and z from the replay buffer and re-compute the (cross-)entropy terms. Here, $\varphi^{'}$ denotes the parameters of a target Q-network (Mnih et al., 2015) that we occasionally copy from the current estimate φ and c_k are the per-step importance weights. Further, we bootstrap the infinite sum after N-steps with $\mathbb{E}_{\pi}\left[Q^{\pi}_{\varphi'}(s_N, .; z_N, \tau)\right]$ instead of introducing a λ parameter as in the original paper. Equipped with this Q-function, we can update the policy and embedding network parameters without requiring additional environment interactions by optimizing the same objective as Eq. 4.

We highlight that the above derivation and steps also hold when the task id is constant, i.e., for the regular reinforcement learning setting rather than the meta-reinforcement learning setting. The following sections in Appendix C present a more detailed theoretical analysis of our adversarial network training regime, essentially showing that the training criterion converges appropriately given the necessary conditions are guaranteed. In practice, we cannot sequentially train the two networks. We must implement the game using an iterative back and forth approach. Optimizing $\mathcal{E}$ to completion would lead to overfitting, and the Encoder would not learn the adequate skill embedding to be used by the policy network. Instead, we alternate between k steps of optimizing the protagonist and one stage of optimizing the adversary. This results in the protagonist being maintained near its optimal solution, so long as adversarial skills change slowly enough. It might seem counter-intuitive to minimize the discounted rewards by the embedding function. However, as long as the hyperparameter α is set correctly, the Jensen-Shannon Divergence objective prioritizes the discounted rewards criterion. With this assumption, we can ensure that minimizing the expected returns would merely act as an implicit noise (Hjelm et al., 2018).

B Variational Bound Inference.

We borrow ideas from variational inference literature to introduce an information-theoretical regularization that encourages versatile skills. In particular, we present a lower entropy of marginal entropy $\mathcal{H}[p(x)]$, which will prove helpful when applied to our objective function from Sec. 3. Note that this section is not novel and is only provided to help readers better understand our proposed methodology.

Proposition B.1. *The lower bound on the marginal entropy $\mathcal{H}[p(x)]$ corresponds to:*

$$\mathcal{H}[p(x)] \geq \int\int p(x, z)\log\left(\frac{q(z|x)}{p(x, z)}dz\right)dx, \tag{11}$$

where $q(z|x)$ is the variational posterior

Proof.

$$
\begin{aligned}
H[p(x)] &= \int -p(x)\log[p(x)]dx = \int p(x)\log\left(\int q(z|x)\frac{1}{p(x)}dz\right)dx \\
&= \int p(x)\log\left(q(z|x)\frac{p(z|x)}{p(x,z)}dz\right)dx \geq \int p(x)\int p(z|x)\log\left(\frac{q(z|x)}{p(x,z)}dz\right)dx \\
&= \int\int p(x,z)\log\left(\frac{q(z|x)}{p(x,z)}dz\right)dx
\end{aligned}
\tag{12}
$$

From the above Equation 12, we can construct a lower bound for our entropy term $\mathcal{H}[\pi(a_i|s_i,t)]$ as follows:

$$
\begin{aligned}
H[\pi(a|s,t)] \geq \mathbb{E}_{\pi_\theta(a,z|s,t)}&\left[\log\left(\frac{q(z|a,s,t)}{\pi(a,z|s,t)}\right)\right] \\
&= \int\int p(\pi_\theta(a,z|s,t))\log\left(\frac{q(z|a,s,t)}{\pi(a,z|s,t)}\right)\partial a\partial z \\
&= \int\int p(z|a,s,t)\pi(a|s,t)\log\left(\frac{q(z|a,s,t)}{\pi(a,z|s,t)}\right)\partial a\partial z \\
&= \int\int p(z|a,s,t)\pi(a|s,t)\left[\log\left(q(z|a,s,t)\right) - \log\left(\pi(a,z|s,t)\right)\right]\partial a\partial z \\
&= \int\int \pi(a|s,t)p(z|a,s,t)\log\left(q(z|a,s,t)\right)\partial a\partial z \\
&\quad - \int\int \pi(a|s,t)p(z|a,s,t)\log\left(\pi(a,z|s,t)\right)\partial a\partial z \\
&= -\int \pi(a|s,t)\mathcal{CE}[p(z|a,s,t)||q(z|a,s,t)]\partial a \\
&\quad - \int\int \pi(a|s,t)p(z|a,s,t)\log\left(\pi(a,z|s,t)\right)\partial a\partial z \\
&= -\mathbb{E}_{\pi(a|s,t)}\left[\mathcal{CE}[p(z|a,s,t)||q(z|a,s,t)]\right] \\
&\quad - \int\int \pi(a|s,t)p(z|a,s,t)\log\left(\pi(a,z|s,t)\right)\partial a\partial z
\end{aligned}
\tag{13}
$$

Where $\mathcal{CE}$ is cross entropy. To simplify the second part of the result from Equation 13, we simplify it as follows:

$$
\begin{aligned}
&= -\int\int \pi(a|s,t)p(z|a,s,t)\log\left(\pi(a,z|s,t)\right)\partial a\partial z \\
&= -\int\int \left[\pi(a|s,t)p(z|a,s,t)\log\left(p(z|s,t)\right) + \pi(a|s,t)p(z|a,s,t)\log\left(\pi(a|s,t,z)\right)\right]\partial a\partial z \\
&= -\int\int p(z,a|s,t)\log\left(p(z|s,t)\right)\partial a\partial z - \int\int \pi(a|s,t)p(z|a,s,t)\log\left(\pi(a|s,t,z)\right)\partial a\partial z \\
&= -\int\int p(z,a|s,t)\log\left(p(z|s,t)\right)\partial a\partial z - \int\int p(z,a|s,t)\log\left(\pi(a|s,t,z)\right)\partial a\partial z \\
&= -\int p(z|s,t)\log\left(p(z|s,t)\right)\partial z - \int\int p(z|s,t)\pi(a|s,t,z)\log\left(\pi(a|s,t,z)\right)\partial a\partial z
\end{aligned}
\tag{14}
$$

Since, the skill embedding is conditionally independent of the state of the agent given task t, we can simplify $p(z|s,t)$ as $p(z|t)$. Similarly, since the action a is conditionally independent of the task t given the latent z, we can simplify $\pi(a|s,t,z)$ as $\pi(a|s,z)$. This is possible since z carries all the necessary information from t, which is required to solve the task. With the above simplifications, Equation 14 further simplifies to:

$$\begin{aligned}
&= -\int p(z|s,t)\log\left(p(z|s,t)\right)\partial z - \int\int p(z|s,t)\pi(a|s,t,z)\log\left(\pi(a|s,t,z)\right)\partial a\partial z \\
&= -\int p(z|t)\log\left(p(z|t)\right)\partial z - \int\int p(z|t)\pi(a|s,z)\log\left(\pi(a|s,z)\right)\partial a\partial z \\
&= \mathcal{H}[p(z|t)] + \int p(z|t)\mathcal{H}\left[\pi(a|s,z)\right]\partial z \\
&= \mathcal{H}[p(z|t)] + \mathbb{E}_{p(z|t)}\left[\mathcal{H}\left[\pi(a|s,z)\right]\right]
\end{aligned} \tag{15}$$

Note that $q(z|a,s,t)$ is the variational inference distribution we are free to choose. Since $q(z|a,s,t)$ is intractable, we resort to a sample-based evaluation of the Cross-Entropy term. This bound holds for any q. Similar to Hausman et al. (2018), we avoid conditioning q on task t to ensure that a given trajectory alone will allow us to identify its skill embedding. Substituting the results from Equation 14, Equation 15, we can rewrite Equation 13 as:

$$\begin{aligned}
H[\pi(a|s,t)] =& -\mathbb{E}_{\pi(a|s,t)}\left[\mathcal{CE}[p(z|a,s,t)||q(z|a,s,t)]\right] + \mathcal{H}[p(z|t)] + \mathbb{E}_{p(z|t)}\left[\mathcal{H}\left[\pi(a|s,z)\right]\right] \\
=& -\int \pi(a|s,t)p(z|a,s,t)\log\left[q(z|a,s)\right]\partial a + \mathcal{H}[p(z|t)] + \mathbb{E}_{p(z|t)}\left[\mathcal{H}\left[\pi(a|s,z)\right]\right] \\
=& \int p(a,z|s,t)\log\left[q(z|a,s)\right]\partial a + \mathcal{H}[p(z|t)] + \mathbb{E}_{p(z|t)}\left[\mathcal{H}\left[\pi(a|s,z)\right]\right] \\
=& \mathbb{E}_{p(a,z|s,t)}\left[\log\left[q(z|a,s)\right]\right] + \mathcal{H}[p(z|t)] + \mathbb{E}_{p(z|t)}\left[\mathcal{H}\left[\pi(a|s,z)\right]\right]
\end{aligned} \tag{16}$$

□

C Theoretical Results

We will first prove that the minimax game has a global optimum for a given policy network and a given embedding network in the case of an on-policy setting, where the optimal policy is already known to us. This proof is presented in Appendix C.1.

These proofs guarantee the working of our algorithm and set the necessary conditions for the model to train appropriately.

C.1 On-Policy Optimality

Proposition C.1. *For a given $\mathbb{E}$, the optimal policy π is*

$$\pi^*_{\mathbb{E}}(\tau) = arg\,max_{\pi} Q^{\varphi}_{\pi}(s,a,z,\tau)$$

and the bound on $Q^{\varphi}_{\pi}(s,a,z,\tau)$ is going to be such that:

$$Q^{\varphi}_{\pi}(s,a,z,\tau) \leq \frac{R_{\max}}{1-\gamma} + \alpha_3\frac{\log|a_{\max}|}{1-\gamma}$$

Proof. The training criterion for the given policy network π, given any encoder $\mathcal{E}$, is to maximize the quantity $V(\mathcal{E},\pi)$ ($-\mathcal{L}_{\text{pro}}$) from Equation 1. Our equation now reduces to:

$$\begin{aligned}
V(\mathbb{E},\pi) =& \max_{\pi} Q^{\varphi}_{\pi}(s,a,z,\tau) \\
=& \max_{\pi}\sum_{i=0}^{\infty}\gamma^i\left[r_{\tau}(s_i,a_i) + \alpha_2\log[q(z|a_i,s_i^H)] + \alpha_3\mathcal{H}[\pi(a|s,z)]\right]
\end{aligned} \tag{17}$$

Here, we will assume the rewards to be bounded by the range $[0, R_{\max}]$. Note that any reward function, sparse or otherwise, can be transformed to the above range and subject to the same proof.

Furthermore, we assume perfect optimality of the inference network q, and the resulting Cross Entropy loss can be omitted since it would be 0. With the use of theory from Kullback–Leibler divergence and Shannon Entropy, we know that $\mathcal{H}(x) \leq log\,|x|$. In our case, the entropy $\mathcal{H}[\pi(a|s,z)]$ can be bounded to $\log |a_t|$, where a_τ is the action space of the given task. For simplification, we approximate $|a_t|$ to $|a_{\max}|$, where $a_{\max}$ denotes the action space with maximum cardinality. We can further simplify the equation above as follows:

$$\begin{aligned} Q^{\varphi}_{\pi}(s,a,z,\tau) &= \sum_{i=0}^{\infty} \gamma^i \left[r_\tau(s_i,a_i) + \alpha_2 \log[q(z|a_i, s_i^H)] + \alpha_3 \mathcal{H}[\pi(a|s,z)] \right] \\ &\leq \sum_{i=0}^{\infty} \gamma^i \left[R_{\max} + \alpha_3 \log |a_{\max}| \right] \\ &\leq \frac{R_{\max}}{1-\gamma} + \alpha_3 \frac{\log |a_{\max}|}{1-\gamma} \end{aligned} \tag{18}$$

The Equation 4 can now be reformulated as:

$$\begin{aligned} C(\mathbb{E}) &= \min_{\mathcal{E}} \mathcal{L}_{\text{adv}} \\ &= \frac{R_{\max}}{1-\gamma} + \alpha_3 \frac{\log |a_{\max}|}{1-\gamma} - \alpha \mathcal{JSD}(z,\tau) \end{aligned} \tag{19}$$

□

Proposition C.2. *The global minimum of the training criterion $C(\mathcal{E})$ is achieved if and only if*

$$\alpha > \frac{R_{\max}}{1-\gamma} + \alpha_3 \frac{\log |a_{\max}|}{1-\gamma} \tag{20}$$

At this point, $C(\mathbb{E})$ achieves a minimum value bounded by $C(\mathbb{E}) < 0$ and is not a trivial solution where the embedding function is an identity function. Furthermore, the property that z and t are mutually exclusive is held.

Proof. Since the Jensen–Shannon divergence between two distributions is always non-negative and zero if they are equal. Since we want to minimize Equation 19, we escape the trivial solution where the embedding network is an identity function. Note that, since the input to the embedding network is a one-hot encoded embedding of the task id, it satisfies the condition of the sharp peak and is uniform in terms of tasks and skills. For the time being, let us assume α is high enough and the $\mathcal{JSD}$ objective is being tuned rather than the term that helps deceive the agent network. In this scenario, we have two unique cases: (i) when $z \sim \tau$, and when (ii) $z \not\sim t$. Note that the Jensen-Shannon divergence would be equal to 0 in the first case and positive in the second. Since our goal is to minimize the objective, the model would converge towards the goal where $z \not\sim \tau$, and escape the trivial solution of the identity function. However, to ensure that the $\mathcal{JSD}$ objective is being optimized with higher precedence, we set α such that:

$$\alpha \mathcal{JSD}(z,\tau) > \frac{R_{\max}}{1-\gamma} + \alpha_3 \frac{\log |a_{\max}|}{1-\gamma} \tag{21}$$

Since we have already shown that the optimal solution is when $z \not\sim \tau$, or when z and τ are mutually exclusive, and JSD would be at its maximum value of 1. This mutual exclusivity property is essential since we would like the same skill to be shared across tasks, as long as the tasks are based on similar structures. For instance, turning a doorknob or screwing a cap on a bottle involves identical skills. Using these fundamental skills to improve task structure in meta-learning would facilitate more accessible posterior adaptation and causal inference. With the assurance that our intuition is following the mathematical rigor, we can select α such that:

$$\alpha > \frac{R_{\max}}{1-\gamma} + \alpha_3 \frac{\log |a_{\max}|}{1-\gamma} \tag{22}$$

Provided this condition is met, the training criterion $C(E)$ achieves a minimum value bounded such that $C(\mathbb{E}) < 0$. We select hyperparameters (α_3, α) such that the following condition is satisfied. □

D Tractable optimization of objective

Here, we show that the equation presented in Equation 4 might not be the ideal form to compute the loss. This is because the computation of the Jensen-Shanon Divergence forces the shape of both skills z and task τ to be the same. Furthermore, since z is derived from τ, it is not entirely straightforward how to compute $\mathcal{H}(\tau|z)$. To subvert these issues, we simplify the above equation as follows.

Recall,

$$\mathcal{H}(\tau) - \mathcal{H}(\tau|z) = \mathcal{H}(z) - \mathcal{H}(z|\tau) \tag{23}$$

Note that our original objective equation does have the L.H.S. term. Thus, we simplify the objective to:

We simplify $H(\tau|z)$ as $H(\tau) - H(z) + H(z|\tau)$. Note that the $H(\tau)$ term cancels out and we are left with $2\alpha(H(z) - H(z|\tau))$ as our second term in Equation 4. This simplification is what we use while computing, but we use the Jensen-Shannon Divergence simplification from Equation 4 to complete our proof of convergence of the model.

$$\begin{aligned}\mathcal{L}_{\text{adv}} &= [Q_\pi^\varphi(s,a;z,\tau)] - \alpha\mathbb{E}_{\tau\in\mathcal{T}}\left[\mathcal{H}(z) - \mathcal{H}(z|\tau) + \mathcal{H}(\tau) - \mathcal{H}(\tau|z)\right] \\ &= [Q_\pi^\varphi(s,a;z,\tau)] - \alpha\mathbb{E}_{\tau\in\mathcal{T}}\left[\mathcal{H}(z) - \mathcal{H}(z|\tau) + \mathcal{H}(z) - \mathcal{H}(z|\tau)\right] \\ &= [Q_\pi^\varphi(s,a;z,\tau)] - 2*\alpha\mathbb{E}_{\tau\in\mathcal{T}}\left[\mathcal{H}(z) - \mathcal{H}(z|\tau)\right]\end{aligned} \tag{24}$$

Since α is a hyperparameter of our choice, we can assign 2α as α' and simplify the equation as follows:

$$\mathcal{L}_{\text{adv}} = [Q_\pi^\varphi(s,a;z,\tau)] - \alpha'\mathbb{E}_{\tau\in\mathcal{T}}\left[\mathcal{H}(z) - \mathcal{H}(z|\tau)\right] \tag{25}$$

The above equation is what we use to make the computation tractable.

E Limitations of Adversarial Training Regime

As discussed briefly in this section, working in an adversarial training regime does pose its own set of limitations. If the hyperparameters are not carefully set, it might become unstable and inconsistent when the adversary completely nullifies the learning from the protagonist. This could occur if the conditions highlighted in Appendix C are not met. This could also happen if the number of iterations the adversary is trained is much greater than the number of iterations of the protagonist forcing the final weights to be closer to an agent trying to minimize the rewards instead of maximizing. Furthermore, setting α in Equation 2 very low will give more importance to learning skills that fool the agent but are not necessarily diverse and distinct. Although our adversarial framework does outperform the traditional training regime in multiple environments, it isn't easy to support the need for such a framework due to its inherent counter-intuitiveness. It requires more theoretical studies to understand this behavior.

F Embedding Efficiency

We want our learned skills to be efficient and diverse in the latent embedding space. To evaluate the efficiency of our embeddings, we compute the diversity of these latents as the volume parallelopiped. Suppose our embedding vector is $z \sim \mathbb{R}^n$. If the volume the skills encompass is higher, more volume of the $\mathbb{R}^n$ space it covers is higher, and we can better adapt to hierarchical skills with the same embedding space. With the given intuition, we compute the efficiency of the embedding as the square of the volume parallelopiped by the embedding. Appendix F.1 discusses our approach to computing this efficiency in more detail. Our experiments show that the latent skills learned from the ATE-PPO algorithm are more efficient and diverse than the latent skills learned from the TE-PPO algorithm. We summarize our findings in Table 2. Furthermore, when computing the volume of the space it covers, we can scale each latent feature by a constant, if needed, without any loss of information.

Table 2: Embedding Efficiency of learned latents

ENVIRONMENT	TE-PPO	ATE-PPO
POINTMASS	$5.88e^{-15}$	$\mathbf{1.87e^{-2}}$
2D NAVIGATION	$1.97e^{-1}$	$\mathbf{8.13e^{-1}}$
MT5	$4.99e^{-16}$	$\mathbf{8.95\ e^{-8}}$

F.1 COMPUTING VOLUME OF LATENT EMBEDDING

Proposition F.1. *Let Π be an m-dimensional parallelotope defined by edge vectors $\mathcal{B} = \{v_1, v_2, ..., v_m\}$, where $v_i \in \mathbb{R}^n$ for $n \geq m$. That is, we are looking at an m-dimensional parallelotope embedded inside n-dimensional space. Suppose $\mathcal{A}$ is the $m \times n$ matrix with row vectors $\mathcal{B}$ given by:*

$$\mathcal{A} = \begin{pmatrix} v_1^T \\ \vdots \\ v_m^T \end{pmatrix}$$

Then the m-dimensional volume of the paralleletope is given by:

$$[vol(\pi)]^2 = \det(AA^T)$$

Proof. Note that AA^T is an $m \times m$ square matrix. Suppose that $m = 1$, then:

$$\det(AA^T) = \det(v_1 v_1^T) = v_1 \cdot v_1 = \|v_1\|^2 = [\text{vol}_1(v_1)]^2$$

so the proposition holds for $m = 1$. From this base equation, we prove the above theorem by induction. Now, we induct on m.

Let us assume the proposition holds for $m^{'}$ such $m^{'} \geq 1$. If we can also prove that the proposition holds for $m^{'} + 1$, we would have proved the above theorem. Letting $A_{m'}$ denote the matrix containing rows v_1 to $v_{m'}$, we can write $A = A_{m'+1}$ as:

$$A = \begin{pmatrix} A_{m'} \\ v_{m'+1}^T \end{pmatrix}$$

We may decompose $v_{m'+1}$ orthogonally as:

$$v_{m'+1} = v_\perp + v_\|$$

where $v_\perp$ lies in the orthogonally complement of the base (i.e., the height of our parallelepipe), and $v_\perp \cdot v_i = 0, \forall\ 1 \leq i \leq m^{'}$. Furthermore, $v_\|$ must be in the span of vectors $\{v_1, v_2, ..., v_{m'}\}$, such that:

$$v_\| = c_1 v_1 + ... c_{m'} v_{m'}$$

We apply a sequence of elementary row operations to A, adding a multiple $-c_i$ of row i to row $m^{'} + 1, \forall\ 1 \leq i \leq m^{'}$. We can then write the resulting matrix B as:

$$B = \begin{pmatrix} A_{m'} \\ \\ v_\perp^T \end{pmatrix} = E_{m'} ... E_1 A,$$

Each E_i is an elementary matrix adding a multiple of one row to another. Notice that the above operation corresponds to shearing the parallelotope so that the last edge is perpendicular to the base. We see that these operations do not change the determinant as:

$$\det(BB^T) = \det(E_{m'} ... E_1 (AA^T) E_1^T ... E_{m'}^T) = \det(AA^T)$$

Through block multiplication, we can obtain BB^T as follows:

$$\begin{aligned} BB^T &= \begin{pmatrix} A_{m'} \\ v_\perp^T \end{pmatrix} \begin{pmatrix} A^T_{m'} & v_\perp \end{pmatrix} \\ &= \begin{pmatrix} A_{m'} A^T_{m'} & A_{m'} v_\perp \\ v_\perp^T A^T_{m'} & v_\perp^T v_\perp \end{pmatrix} \\ &= \begin{pmatrix} A_{m'} A^T_{m'} & A_{m'} v_\perp \\ (A_{m'} v_\perp)^T & \|v_\perp\|^2 \end{pmatrix} \end{aligned}$$

Furthermore, notice that

$$A_{m'} v_\perp = \begin{pmatrix} v_1^T \\ \vdots \\ v_{m'}^T \end{pmatrix} v_\perp = 0$$

Therefore, we have

$$BB^T = \begin{pmatrix} A_{m'} A^T_{m'} & 0 \\ 0^T & \|v_\perp\|^2 \end{pmatrix}$$

Taking the determinant, we can simplify $\det(BB^T)$ as

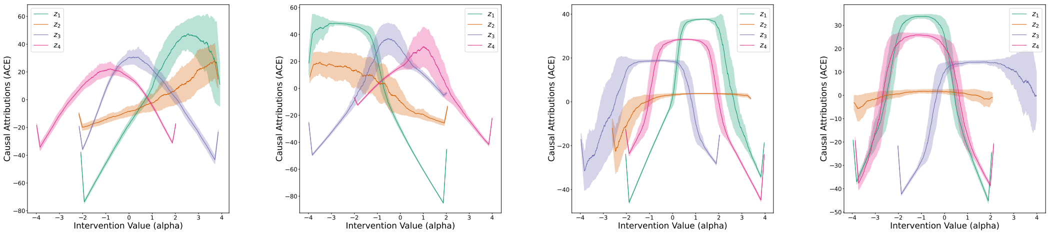
$$\det(BB^T) = \|v_\perp\|^2 \det(A_{m'} A^T_{m'})$$

By definition, $\|v_\perp\|$ is the height of the parallelotope, and by the induction hypothesis, $\det(A_{m'}, A^T_{m'})$ is the square of the base. Therefore, we have proved the above theorem by induction. □

G Analysis of Skills Learned

In this section, we discuss a few additional results, along with a study of the effect of each latent feature on the agent's behavior.

G.1 PointMass Environment

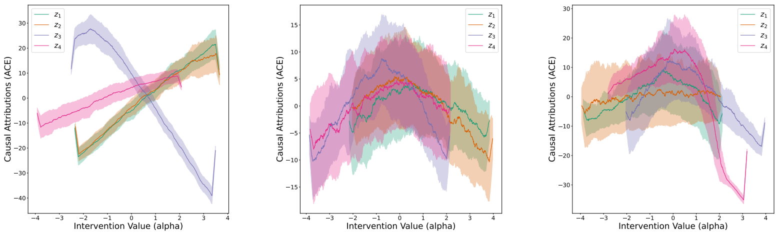


(a) Goal: Left (b) Goal: Right (c) Goal: Top (d) Goal: Bottom

Figure 6: The Average Causal Effect of the latent skills for different tasks in the PointMass Environment. From the above plots, we can make a few inferences, such as the fact that embedding z_2 seems to be an auxiliary feature and does not play a significant role in the behavior of the agent.

From the ACE plots, we can successfully assert that feature z_2 is an auxiliary variable and does not play any significant role in the model's behavior since the ACE value is close to 0. Furthermore, we notice that the z_3 and z_4 causal analysis are pretty similar, and z_1 seems to be an essential feature in the pool of latent features.

To better understand the effect of each feature on the behavior of the model, we use an input perturbation method. Here we fix all features to a given task: say Top, but vary a given feature within the skill to study the change in the agent's behavior. Figure 11 depicts the results from our input perturbation experiment.



(a) Goal: Right (b) Goal: Top (c) Goal: Bottom

Figure 7: The Average Causal Effect of the latent skills for different tasks in the 2-D Navigation Environment. From the above plots, we notice that there is a lot of overlap and noise in the ACE, especially in the case of Figure 7(b) and Figure 7(c). This is expected since the model needs to learn to diverge from its initial path after a few time steps, bringing exploration only after it crosses the corridor. We hypothesize that we do not observe the same noise in Figure 7(a) since there is no divergence from the path.

G.2 2-D Navigation Environment

The average importance of each feature is shown in Figure 8.

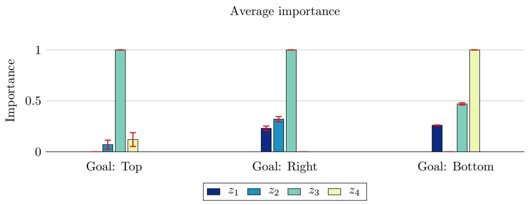
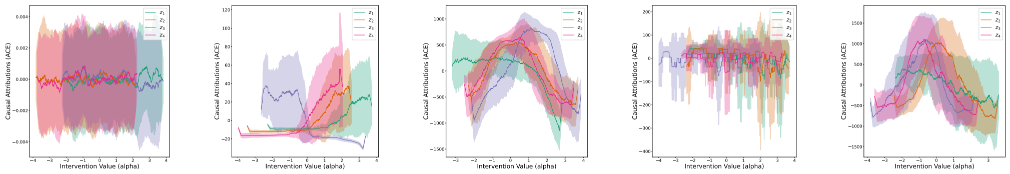


Figure 8: The normalized importance of each latent feature on the 2-D navigation tasks across 5 different seeds.

From the ACE plots, we can successfully assert that feature z_2 is an auxiliary variable and does not play any significant role in the model's behavior since the ACE value is close to 0. Furthermore, we notice that the z_3 and z_4 causal analysis are pretty similar, and z_1 seems to be an essential feature in the pool of latent features.

To better understand the effect of each feature on the behavior of the model, we use an input perturbation method. Here we fix all features to a given task: say Top, but vary a given feature within the skill to study the change in the agent's behavior. Figure 12 depicts the results from our input perturbation experiment.

G.3 Meta-World(MT5) Environment



(a) Push (b) Window Open (c) Window Close (d) Drawer Open (e) Drawer Close

Figure 9: The Average Causal Effect of the latent skills for different tasks in the Metaworld (MT5) Environment. From the above plots, we notice that there is a lot of overlap and noise in the ACE. Furthermore, we notice that the ACE of the tasks: Push and Drawer Open, is roughly zero across all latents, and our model has prioritized learning other tasks.

The average importance of each feature is shown in Figure 10.

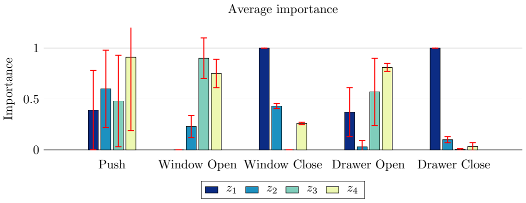
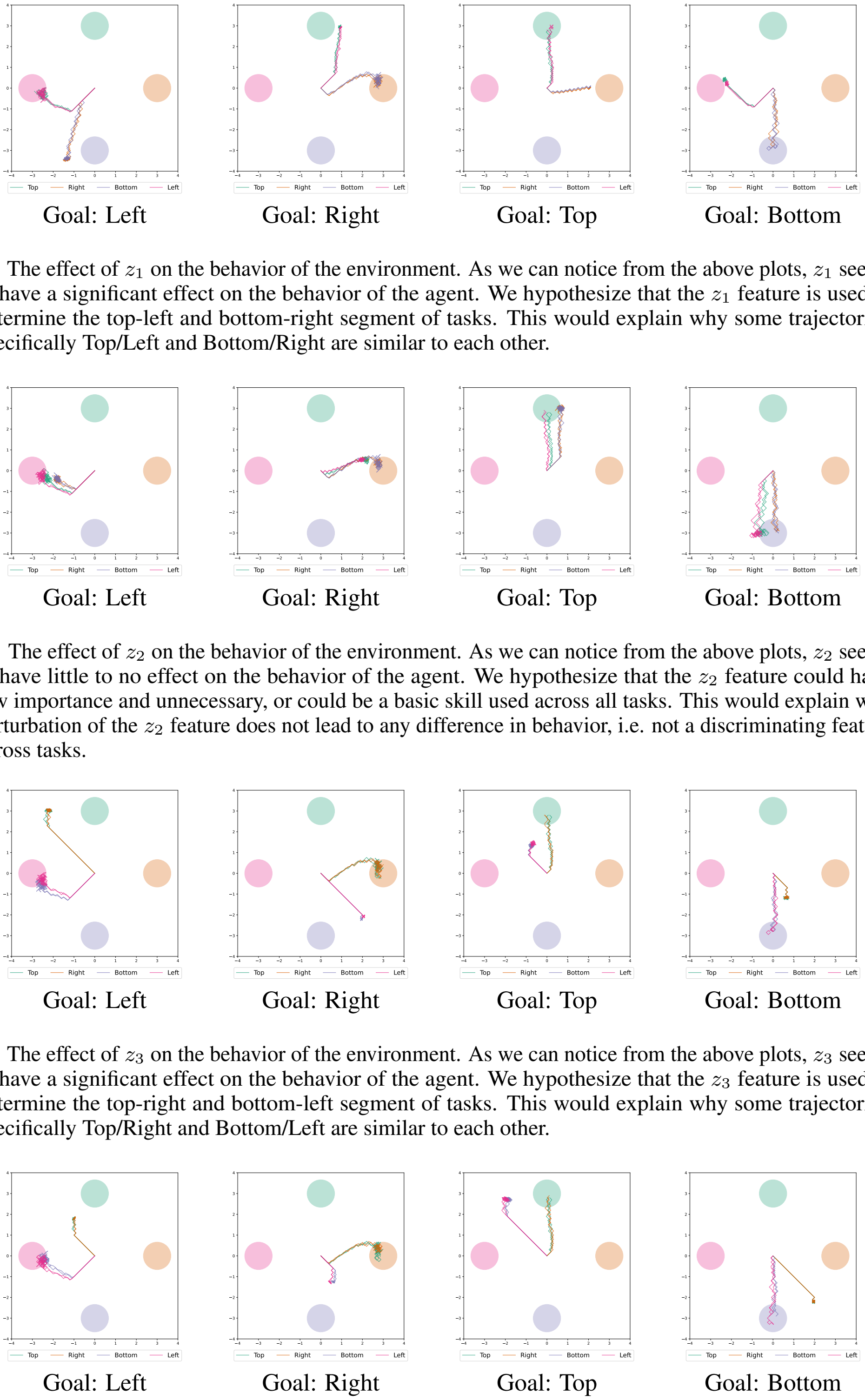


Figure 10: The normalized importance of each latent feature on the Metaworld (MT5) tasks across 5 different seeds.

From the ACE plots, we can successfully assert that feature z_2 is an auxiliary variable and does not play any significant role in the model's behavior since the ACE value is close to 0. Furthermore, we notice that the z_1 variable is significant for tasks that involve closing, whereas z_2 and z_3 variable are significant when tasks involve the action of opening.

Performing an input-perturbation method, similar to previous sections might not be feasible for the MetaWorld environment, as it involves much higher complexity of observation and action space for a simple visualization.



Goal: Left Goal: Right Goal: Top Goal: Bottom

(a) The effect of z_1 on the behavior of the environment. As we can notice from the above plots, z_1 seems to have a significant effect on the behavior of the agent. We hypothesize that the z_1 feature is used to determine the top-left and bottom-right segment of tasks. This would explain why some trajectories, specifically Top/Left and Bottom/Right are similar to each other.

Goal: Left Goal: Right Goal: Top Goal: Bottom

(b) The effect of z_2 on the behavior of the environment. As we can notice from the above plots, z_2 seems to have little to no effect on the behavior of the agent. We hypothesize that the z_2 feature could have low importance and unnecessary, or could be a basic skill used across all tasks. This would explain why perturbation of the z_2 feature does not lead to any difference in behavior, i.e. not a discriminating feature across tasks.

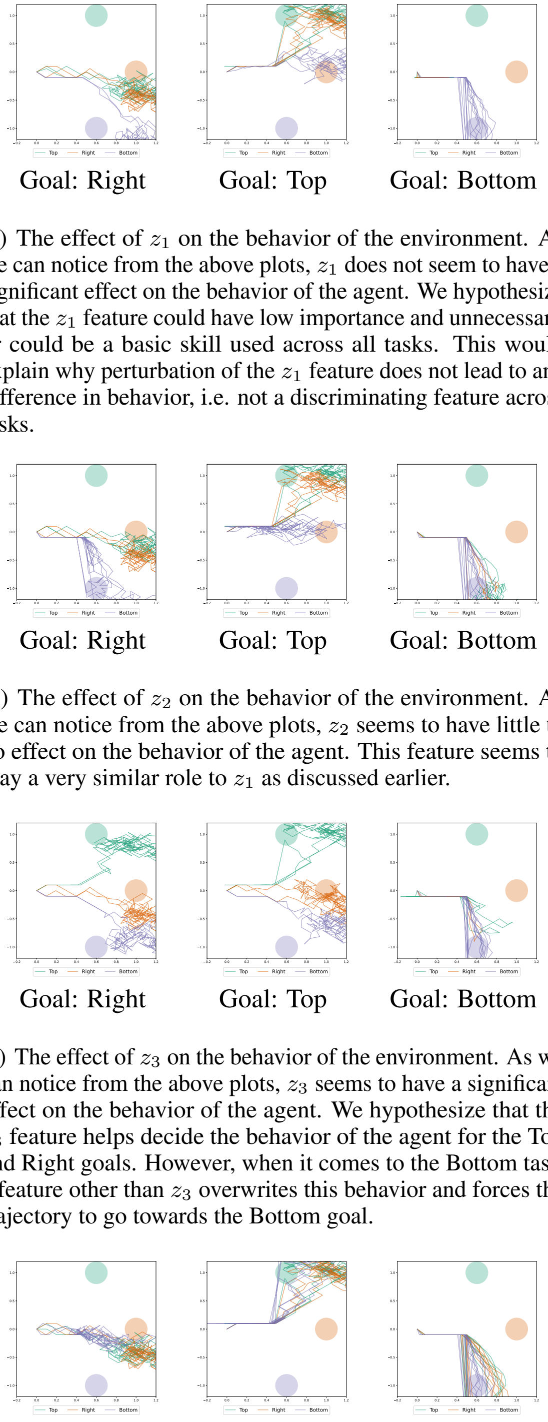
Goal: Left Goal: Right Goal: Top Goal: Bottom

(c) The effect of z_3 on the behavior of the environment. As we can notice from the above plots, z_3 seems to have a significant effect on the behavior of the agent. We hypothesize that the z_3 feature is used to determine the top-right and bottom-left segment of tasks. This would explain why some trajectories, specifically Top/Right and Bottom/Left are similar to each other.

Goal: Left Goal: Right Goal: Top Goal: Bottom

(d) The effect of z_4 on the behavior of the environment. As we can notice from the above plots, z_4 seems to have a similar purpose as the z_3 feature. However, we notice a slight difference in the trajectory - z_4 enforces a shorter distance before the sudden turn in Left/Right, but a longer one in Top/Bottom. Whereas, z_3 does the exact opposite.

Figure 11: Input perturbation results on PointMass environment



Goal: Right Goal: Top Goal: Bottom

(a) The effect of z_1 on the behavior of the environment. As we can notice from the above plots, z_1 does not seem to have a significant effect on the behavior of the agent. We hypothesize that the z_1 feature could have low importance and unnecessary, or could be a basic skill used across all tasks. This would explain why perturbation of the z_1 feature does not lead to any difference in behavior, i.e. not a discriminating feature across tasks.

Goal: Right Goal: Top Goal: Bottom

(b) The effect of z_2 on the behavior of the environment. As we can notice from the above plots, z_2 seems to have little to no effect on the behavior of the agent. This feature seems to play a very similar role to z_1 as discussed earlier.

Goal: Right Goal: Top Goal: Bottom

(c) The effect of z_3 on the behavior of the environment. As we can notice from the above plots, z_3 seems to have a significant effect on the behavior of the agent. We hypothesize that the z_3 feature helps decide the behavior of the agent for the Top and Right goals. However, when it comes to the Bottom task, a feature other than z_3 overwrites this behavior and forces the trajectory to go towards the Bottom goal.

Goal: Right Goal: Top Goal: Bottom

(d) The effect of z_4 on the behavior of the environment. As we can notice from the above plots, z_4 seems to be the only feature that helps propel the trajectory towards the Bottom goal. This is the feature that overwrites the effect of z_3 feature and forces the action to be towards the destined goal.

Figure 12: Input perturbation results on 2-D Navigation environment

H Model and Hyperparameters

This section discusses the model and hyperparameters used in our experiments.

Table 3 depicts the hyperparameters we used for training our TE-PPO and ATE-PPO algorithm on the PointMass environment.

Similarly, Table 4 depicts the hyperparameters we used for training our TE-PPO and ATE-PPO algorithms in the 2-D navigation environment.

Finally, Table 5 depicts the hyperparameters we used for training our TE-PPO and ATE-PPO algorithm in the Meta-World (MT5) environment.

Table 3: Hyperparameters used for training TE-PPO and ATE-PPO on PointMass environment.

Description	**TE-PPO**	**ATE-PPO**	`argument_name`
General Hyperparameters			
Discount	0.99	0.99	`discount`
Batch size	4096	4096	`batch_size`
Number of epochs	600	600	`n_epochs`
Algorithm-Specific Hyperparameters			
Encoder hidden sizes	$(20, 20)$	$(20, 20)$	`enc_hidden_sizes`
Inference hidden sizes	$(20, 20)$	$(20, 20)$	`inf_hidden_sizes`
Policy hidden sizes	$(32, 16)$	$(32, 16)$	`pol_hidden_sizes`
Activation function of hidden layers	tanh	tanh	`hidden_nonlinearity`
Likelihood ratio clip range	0.2	0.2	`lr_clip_range`
Latent dimension	2	4	`latent_length`
Inference window length	6	6	`inference_window`
Embedding maximum standard deviation	0.2	0.2	`embedding_max_std`
Policy entropy coefficient	$1e^{-3}$	$1e^{-3}$	`policy_ent_coeff`
Encoder entropy coefficient	$1e^{-3}$	$1e^{-3}$	`enc_ent_coeff`
Inference entropy coefficient	$5e^{-2}$	$5e^{-2}$	`inf_ent_coeff`
Optimizer-Specific Hyperparameters			
Protagonist mini-batch size	32	64	`pr_batch_size`
Adversary mini-batch size	–	64	`ad_batch_size`
Inference mini-batch size	32	64	`inf_batch_size`
Protagonist learning rate	$1e^{-4}$	$1e^{-3}$	`pr_lr`
Adversary learning rate	–	$1e^{-4}$	`ad_lr`
Inference learning rate	$1e^{-3}$	$1e^{-3}$	`inf_lr`

Table 4: Hyperparameters used for training TE-PPO and ATE-PPO on 2-D Navigation environment.

Description	**TE-PPO**	**ATE-PPO**	`argument_name`
General Hyperparameters			
Discount	0.99	0.99	`discount`
Batch size	3072	3072	`batch_size`
Number of epochs	400	400	`n_epochs`
Algorithm-Specific Hyperparameters			
Encoder hidden sizes	$(20, 20)$	$(20, 20)$	`enc_hidden_sizes`
Inference hidden sizes	$(20, 20)$	$(20, 20)$	`inf_hidden_sizes`
Policy hidden sizes	$(32, 16)$	$(32, 16)$	`pol_hidden_sizes`
Activation function of hidden layers	tanh	tanh	`hidden_nonlinearity`
Likelihood ratio clip range	0.2	0.2	`lr_clip_range`
Latent dimension	4	4	`latent_length`
Inference window length	6	6	`inference_window`
Embedding maximum standard deviation	0.2	0.2	`embedding_max_std`
Policy entropy coefficient	$1e^{-3}$	$1e^{-3}$	`policy_ent_coeff`
Encoder entropy coefficient	$1e^{-3}$	$1e^{-3}$	`enc_ent_coeff`
Inference entropy coefficient	$5e^{-2}$	$5e^{-2}$	`inf_ent_coeff`
Optimizer-Specific Hyperparameters			
Protagonist mini-batch size	32	64	`pr_batch_size`
Adversary mini-batch size	$-$	32	`ad_batch_size`
Inference mini-batch size	32	64	`inf_batch_size`
Protagonist learning rate	$1e^{-4}$	$5e^{-4}$	`pr_lr`
Adversary learning rate	$-$	$1e^{-4}$	`ad_lr`
Inference learning rate	$1e^{-3}$	$5e^{-4}$	`inf_lr`

Table 5: Hyperparameters used for training TE-PPO and ATE-PPO on MT5 environment.

Description	**TE-PPO**	**ATE-PPO**	`argument_name`
General Hyperparameters			
Discount	0.99	0.99	`discount`
Batch size	25000	25000	`batch_size`
Number of epochs	1000	1000	`n_epochs`
Algorithm-Specific Hyperparameters			
Encoder hidden sizes	$(20, 20)$	$(20, 20)$	`enc_hidden_sizes`
Inference hidden sizes	$(20, 20)$	$(20, 20)$	`inf_hidden_sizes`
Policy hidden sizes	$(32, 16)$	$(32, 16)$	`pol_hidden_sizes`
Activation function of hidden layers	tanh	tanh	`hidden_nonlinearity`
Likelihood ratio clip range	0.2	0.2	`lr_clip_range`
Latent dimension	4	4	`latent_length`
Inference window length	6	6	`inference_window`
Embedding maximum standard deviation	0.2	0.2	`embedding_max_std`
Policy entropy coefficient	$2e^{-}2$	$2e^{-2}$	`policy_ent_coeff`
Encoder entropy coefficient	$2e^{-2}$	$2e^{-2}$	`enc_ent_coeff`
Inference entropy coefficient	$5e^{-2}$	$5e^{-2}$	`inf_ent_coeff`
Optimizer-Specific Hyperparameters			
Protagonist mini-batch size	256	256	`pr_batch_size`
Adversary mini-batch size	−	256	`ad_batch_size`
Inference mini-batch size	256	256	`inf_batch_size`
Protagonist learning rate	$1e^{-3}$	$5e^{-4}$	`pr_lr`
Adversary learning rate	−	$1e^{-4}$	`ad_lr`
Inference learning rate	$1e^{-3}$	$5e^{-4}$	`inf_lr`