AN EMPIRICAL INVESTIGATION OF MUTUAL INFORMATION SKILL LEARNING

Faisal Mohamed
African Institute for Mathematical Sciences
fmohamed@aimsammi.org

Benjamin Eysenbach
Carnegie Mellon University, Google Brain
beysenba@cs.cmu.edu

Ruslan Salakhutdinov
Carnegie Mellon University

ABSTRACT

Unsupervised skill learning methods are a form of unsupervised pre-training for reinforcement learning (RL) that has the potential to improve the sample efficiency of solving downstream tasks. Prior work has proposed several methods for unsupervised skill discovery based on mutual information (MI) objectives, with different methods varying in how this mutual information is estimated and optimized. This paper studies how different design decisions in skill learning algorithms affect the sample efficiency of solving downstream tasks. Our key findings are that the sample efficiency of downstream adaptation under off-policy backbones is better than their on-policy counterparts. In contrast, on-policy backbones result in better state coverage, moreover, regularizing the discriminator gives better downstream results, and careful choice of the mutual information lower bound and the discriminator architecture yields significant improvements in downstream returns, also, we show empirically that the learned representations during the pre-training step correspond to the controllable aspects of the environment[1].

1 INTRODUCTION

Reinforcement learning (RL), when combined with high capacity function approximators (e.g., neural networks), can acquire complex robotic skills in both simulation and the real world (Schulman et al., 2015; Lillicrap et al., 2015; Zhang et al., 2019; Haarnoja et al., 2018a)

However, RL suffers from poor sample efficiency (Schulman et al., 2015; Lillicrap et al., 2015; Zhang et al., 2019; Haarnoja et al., 2018a), the high performance of RL agents comes with the cost of millions of transition data.

Meanwhile, unsupervised pre-training had shown significant improvements in sample efficiency in the fields of natural language processing (NLP) (Brown et al., 2020; Devlin et al., 2019) and computer vision (CV) (Chen et al., 2020; Hénaff et al., 2020).

Combining unsupervised pre-training and RL (unsupervised RL) have shown a promise in improving sample efficiency. (Laskin et al., 2021a; Gregor et al., 2016; Eysenbach et al., 2018), in this paradigm the goal is to train the agent in an interaction phase, in which it can acquire knowledge about the environment and learn useful representations for efficient adaptation to downstream tasks.

Most unsupervised RL algorithms differ in the pre-training objective or the intrinsic reward. Unsupervised RL algorithms are classified into three categories according to their pre-training objective, knowledge-based algorithms Pathak et al. (2017); Burda et al. (2018) maximize a prediction error, while data-based algorithms Yarats et al. (2021); Liu & Abbeel (2021b) optimize a state diversity objective, competence-based algorithms also called unsupervised skill discovery Gregor et al. (2016); Laskin et al. (2021a) maximize the mutual information between the state and skill distributions by applying an off-the-shelf RL algorithm to a self-generated reward function. Skill discovery has shown promising results Laskin et al. (2021b;a); Eysenbach et al. (2018), however, there are many

[1]Code is available at https://github.com/FaisalAhmed0/SLUSD

design decisions that the practitioner needs to decide when applying these algorithms, such as the deep RL optimizer, skills' representation, adding a regularizer, and the mutual information lower bound.

The main contribution of this work is an empirical investigation of these design decisions:

1. We investigate the effect of using different deep RL backbones on the adaptation efficiency and scalability of the skill learning algorithm.
2. We study how various regularization methods affect the learned skills and adaptation efficiency.
3. We study how the choice of the mutual information lower bound affect the quality of the skills.
4. We try to understand the representations learned during the pre-training phase.

2 Related Work

Mutual information (MI) skill learning or competence-based unsupervised RL have been explored by many researchers. In the pre-training phase, all skill learning algorithms have the same MI objective. Algorithms differ in their optimization backbone, the skills representation, the form of the MI lower bound and its approximations among other design choices. There are two ways to express the MI objective, the first one is optimizing the reverse form (Gregor et al., 2016) of mutual information (Eysenbach et al., 2018; Gregor et al., 2016; Hansen et al., 2019; Achiam et al., 2018; Gao et al., 2020; Baumli et al., 2020; Zhao et al., 2021; Florensa et al., 2017) while the second type optimizes the forward form (Laskin et al., 2021a; Liu & Abbeel, 2021a; Lee et al., 2020; Sharma et al., 2020). Campos et al. (2020) designed a skill learning algorithm based on three stages, exploration, skill discovery and skill learning, and used a variational auto-encoder to approximate the MI objective. Laskin et al. (2021b) introduced a unified benchmark for unsupervised RL algorithms and open-sourced code for popular baselines, while Choi et al. (2021) focused on unifying goal-conditioned RL (GCRL) and MI skill learning in one framework that was called variational goal-conditioned RL (VGCRL), and used it to define an evaluation metric for MI skill learning methods.

In this paper, we do not propose a new skill learning algorithm or a benchmark, but we focus on understanding the influence of some important design choices and their effects on the properties of the learned skills.

3 Background

Reinforcement learning. We work under the standard setup of the Markov decision process (MDP) Sutton & Barto (2018), which is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition dynamics (in general $\mathcal{P}$ is a probability distribution but in deterministic environment it is a function), r is the reward function, and γ is the discount factor.

In MDP settings, an agent interacts with an environment by receiving a state at timestep t s_t and choosing an action a_t according to a policy $\pi(a_t \mid s_t)$. After executing the action in the environment, the agent receives a new state s_{t+1} and a reward signal r_{t+1} that indicates how good or bad the executed action is. The goal of the agent is to maximize the expected discounted sum of rewards.

Mutual information lower bounds. Before we formally define the unsupervised RL problem, we review prior work on mutual information estimation(defined in equation 1), which forms the basis for many prior unsupervised RL methods. We will examine different MI lower bounds and their effects on adaptation and the learned skills. We use BA lower bound I_{BA} (Barber & Agakov, 2003), NWJ I_{NWJ} (Nguyen et al., 2010), InfoNCE I_{NCE} (van den Oord et al., 2019), and I_α lower bound (Choi et al., 2021), which are shown on the equations below:

$$I(x;y) = \mathcal{H}(x) - \mathcal{H}(x \mid y) \tag{1}$$
$$= \mathcal{H}(y) - \mathcal{H}(y \mid x)$$
$$I_{BA} = E_{p(x,y)}[\log q(y \mid x)] - \log p(y)] \tag{2}$$
$$I_{NWJ} = E_{p(x,y)}[f(x,y)] - \frac{1}{e} E_{p(x)p(y)}\left[e^{f(x,y)}\right] \tag{3}$$
$$I_{NCE} = E_{p^K(x,y)}\Big[\frac{1}{K}\sum_{i=1}^{K} \log \frac{e^{f(y_i,x_i)}}{\frac{1}{K}\sum_{j=1}^{K} e^{f(y_i,x_j)}}\Big] \tag{4}$$
$$I_\alpha = 1 + \frac{1}{K}\sum_{j=1}^{K}\Big(\frac{1}{K-1}\sum_{i\neq j}\Big(\log \frac{e^{f(x_i,y_i)}}{a(y_i;x_{\neq j})} - \frac{e^{f(x_i,y_j)}}{a(y_i;x_{\neq j})}\Big)\Big) \tag{5}$$

where $a(y; x_{1:M}) = \alpha \frac{1}{M}\sum_{l=1}^{M} e^{f(x_l,y)} + (1-\alpha)q(y)$ and $\mathcal{H}(\cdot)$ is Shannon entropy.

Unsupervised skill learning. We follow the standard unsupervised RL learning paradigm (Gregor et al., 2016; Eysenbach et al., 2018; Laskin et al., 2021b). The agent maximizes a self-generated objective based on an intrinsic reward r_{int}, the goal is to acquire general knowledge and useful representations that improve the data efficiency in the adaptation phase, in which the pre-trained policy is trained to maximize an extrinsic objective based on a task reward r_{ext}.

In MI skill learning, the policy $\pi(a \mid s, z)$ is conditioned on a latent variable $z \sim p(z)$ which we refer to as a "skill", the pre-training objective is defined as the mutual information between the skills and the states.

We use DIAYN (Eysenbach et al., 2018) as our skill-learning method, which optimizes the following lower bound on this mutual information objective:

$$I(s;z) = \mathcal{H}(z) - \mathcal{H}(z \mid s) = E_{z\sim p(z), s\sim p(s)}[\log p(z \mid s)] - E_{z\sim p(z)}[\log p(z)]$$
$$\geq E_{z\sim p(z), s\sim p(s)}[\log q_\phi(z \mid s)] - E_{z\sim p(z)}[\log p(z)] \triangleq J(\theta,\phi) \tag{6}$$

To maximize the expectation in $J(\theta,\phi)$ with a deep RL algorithm, we use the following intrinsic reward in equation7 below.

$$r_{int} = \log q_\phi(z \mid s) - \log p(z) \tag{7}$$

The skill distribution $p(z)$ is fixed to be discrete uniform, $q_\phi(z \mid s)$ is a learned discriminator, we refer the reader to Eysenbach et al. (2018) for a more detailed description of the algorithm.

4 Experiments

We run experiments to investigate the influence of important design decisions on the adaptation efficiency and scalability of the pre-trained policy. In particular, the experiments answer the following questions:

- **Q1** How does a deep RL algorithm affect the speed of adaptation to a downstream task?
- **Q2** Is the mutual information lower bound well correlated with exploration performance ?
- **Q3** Does increasing the pre-training steps improves adaptation efficiency ?
- **Q4** Does regularizing the discriminator lead to better downstream performance?
- **Q5** Do different mutual information lower bounds lead to different performance on downstream tasks?
- **Q6** What do the representations learned during the pre-training step correspond to?

4.1 Experimental Setup

We conducted the experiments on MountainCar, HalfCheetah, Walker2d, and Ant from OpenAI gym Brockman et al. (2016). We chose DIAYN as our skill learning algorithm due to its popularity

as a baseline for mutual information based skill learning. We changed the deep RL optimizer for the first two experiments and fixed the adaptation algorithm to soft actor critic (SAC) Haarnoja et al. (2018b). In the rest of the experiments, we fixed everything and changed other aspects of the algorithm.

4.2 The Optimization Backbone And Downstream Adaptation (Q1, Q2)

We trained DIAYN with two model-free deep RL algorithms. From off-policy methods we chose SAC (as the original implementation) and from on-policy methods, we chose proximal policy optimization (PPO) (Schulman et al., 2017).

In the pre-training phase unlike Laskin et al. (2021b), we trained the policy to convergence (we assumed that the pre-training phase is cheap and we can always train to convergence), to make sure the comparisons are fair and reflects the quality of skills learned by each deep RL backbone, we fixed the algorithm in the adaptation phase, in particular we used SAC with 100K adaptation timesteps.

Table 1 shows the results, and figure 1 shows the learning curves on the downstream task, all results are calculated by training the algorithms on three random seeds (for MountainCar we tried five random seeds but it did not affect the variance significantly), reporting the mean and the standard deviation and normalized by the performance of an expert policy, which was a SAC agent trained on each task with 5M timesteps.

DIAYN with SAC outperformed PPO in downstream adaptation for all environments except Ant. PPO had higher state entropy in most environments, this may indicate that PPO learns a more diverse set of skills with the cost of slightly worse downstream performance, while SAC learns more specialized skills that adapt quickly to some downstream tasks.

In Walker and Ant, RL from scratch (training from randomly initialized weights) was competitive to the pre-trained policies, and the adaption did not improve the pre-trained policy by a considerable amount. This suggests that better adaptation methods should be developed to adapt the learned skills more efficiently, or it may indicate that the skill learning algorithm does not learn useful skills in more complex environments. Hence, better skill learning algorithms should be developed.

Due to the exploration challenge in MountainCar, only DIAYN with SAC succeeded in finding the goal state, which shows that a skill learning algorithm may need an additional exploration strategy or a better approximation of the MI objective to discover skills that have broad coverage of the state space.

In addition, all the downstream returns are far from the state of the art performance(this is also mentioned in Laskin et al. (2021b)), which keep the question of how to develop efficient pre-training methods that reach the state of the art performance with low data open.

There is no clear correlation between state entropy (state diversity) and the intrinsic reward. For example, in MountainCar, DIAYN with SAC had a higher intrinsic reward that results in a higher entropy. But in Walker, the same algorithm had lower entropy despite the higher intrinsic reward. A reason for this could be that the MI approximation (specifically the reverse form) does not incentivize broad coverage of the state space, as indicated by Laskin et al. (2021a).

4.3 The Optimization Backbone And The Algorithm Scalability (Q3)

In this experiment, we investigated the scalability of mutual information skill learning. As mentioned in Laskin et al. (2021b), by scalability, we mean the correlation between the number of interactions in the pre-training phase and the value of the downstream return. The algorithm is scalable if there is a positive correlation between these two quantities.

We trained DIAYN with SAC and PPO and finetuned the pre-trained policy on the downstream performance on fixed checkpoints. The results are shown in figure 2.

The first result to observe is that the two algorithms are not highly scalable, and more pre-training steps do not necessarily result in better downstream performance, at some points it even gives lower returns. When comparing the two algorithms, DIAYN with SAC showed better performance than PPO in all environments, and for most pre-training steps, both algorithms were better than random initialization.

Environment	Algorithm	R_{ext} After Adaptation (%)	R_{ext} Best Skill (%)	R_{int}	State Entropy
MountainCar	SAC	33.15± 58.15	28.62± 60.08	2229.40 ± 10.36	-9.41 ± 0.65
	PPO	-0.00 ± 0.00	-1.52± 1.58	1643.41± 238.91	-9.84 ± 0.38
HalfCheetah	SAC	68.88± 10.25	-1.78± 0.64	3400.37± 0.63	-162.83± 3.87
	PPO	67.66 ± 8.55	4.49 ± 6.74	3276.30 ± 208.87	-33.34 ± 19.39
Walker2d	SAC	5.84 ± 3.38	16.57 ± 1.7	3342.23 ± 64.15	-16.14 ± 12.7
	PPO	5.60 ± 0.77	2.11 ± 2.22	3150.90 ± 223.61	35.89 ± 8.39
Ant	SAC	9.63 ± 12.68	-2.94 ± 3.59	1242.69± 2054.93	-1139.12 ± 31.60
	PPO	9.64 ± 3.12	-0.12 ± 0.03	3337.75 ± 48.78	-846.48 ± 39.26

Table 1: **DIAYN with different Deep RL optimizers:** In most environments, PPO had higher state entropy . This may indicate that PPO learns more diverse skills than SAC. In Walker, adaptation yielded worse results than the best skill return in SAC, and in Ant, SAC did not converge to the highest intrinsic return.

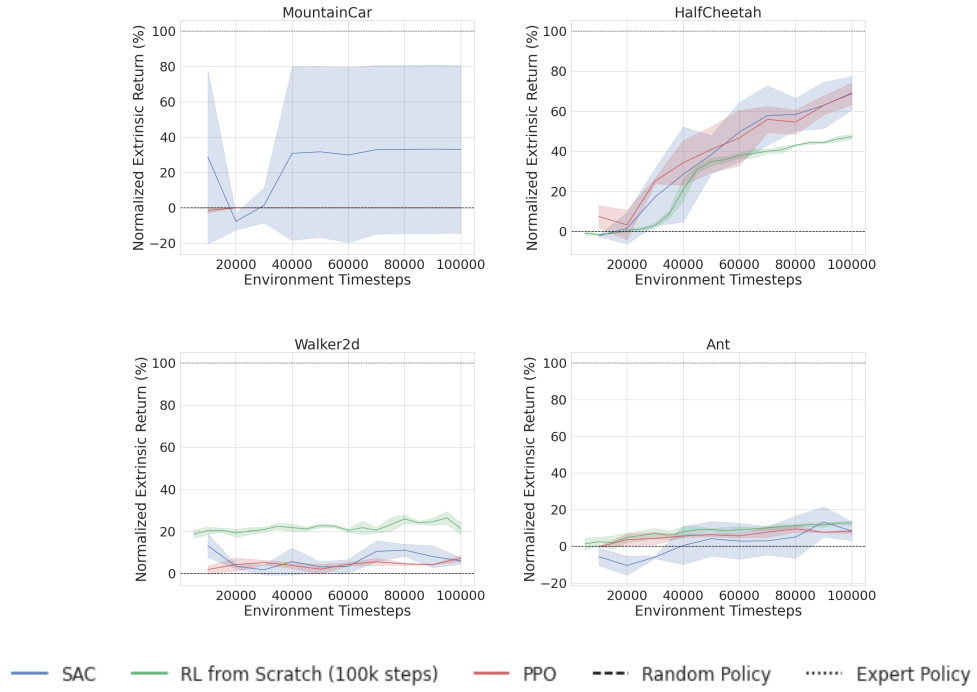

Figure 1: **Downstream adaptation:** In downstream adaptation, DIAYN with SAC outperformed PPO in most environments, due to the exploration challenge in MountainCar, only DIAYN with SAC succeeded in finding the goal state that has the high reward, all the downstream returns are far from the state of the art performance.

This result raises the question of how to improve the scalability of skills learning algorithms. There could be multiple causes of this problem. Maybe we need better adaptation methods, better approximations of the MI objective, or methods that help the pre-trained policy in learning scalable representations.

A possible answer for this may be that the downstream task can not be learned effectively using the learned set of skills, which indicates that we should learn better skill representations that learn more skills as we increase the pre-training interactions, or the skills can be combined to cover tasks beyond what each skill individually does.

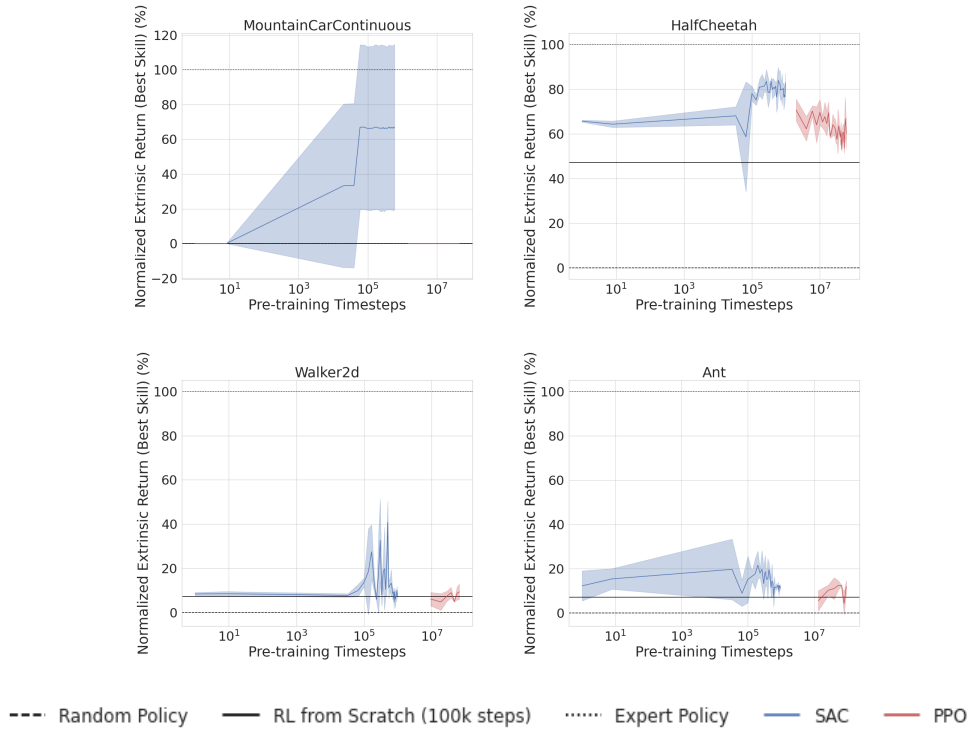

Figure 2: **Scalability with the number of pre-training timesteps:** Both algorithms are not scalable, more pre-training steps do not necessarily result in better downstream performance, and in some steps, it gives lower returns. Both algorithms are better than random initialization.

4.4 Effect of regularization on downstream performance. (Q4)

For a fair comparison, we fixed the deep RL backbone (SAC) and run DIAYN with different regularization methods applied to the discriminator, we trained the discriminator with weight decay, dropout Srivastava et al. (2014), gradient penalty Gulrajani et al. (2017), label smoothing Szegedy et al. (2016), and mixup Zhang et al. (2017). The results are shown in figure 3 and table 2 in the appendix.

In most cases, using regularization improved the results of downstream performance. Weight decay improved the results by an average factor of 2.5, dropout gave nearly the same improvements, and gradient penalties introduced the lowers amount of improvements.

This performance gap between regularized and unregularized skill learning begs several questions, what does the discriminator learn? Does it overfit the initial random data induced by the initially random policy? What if we introduced a lag between the discriminator, the policy and the discriminator update? Will this improve the results in the unregularized case? are MLPs the best architecture for skills discrimination?

These questions suggest that we need to develop discriminators that are less prone to overfitting and focus on distilling the state space to a meaningful set of skills as the pre-trained policy converges. We leave these questions for future work.

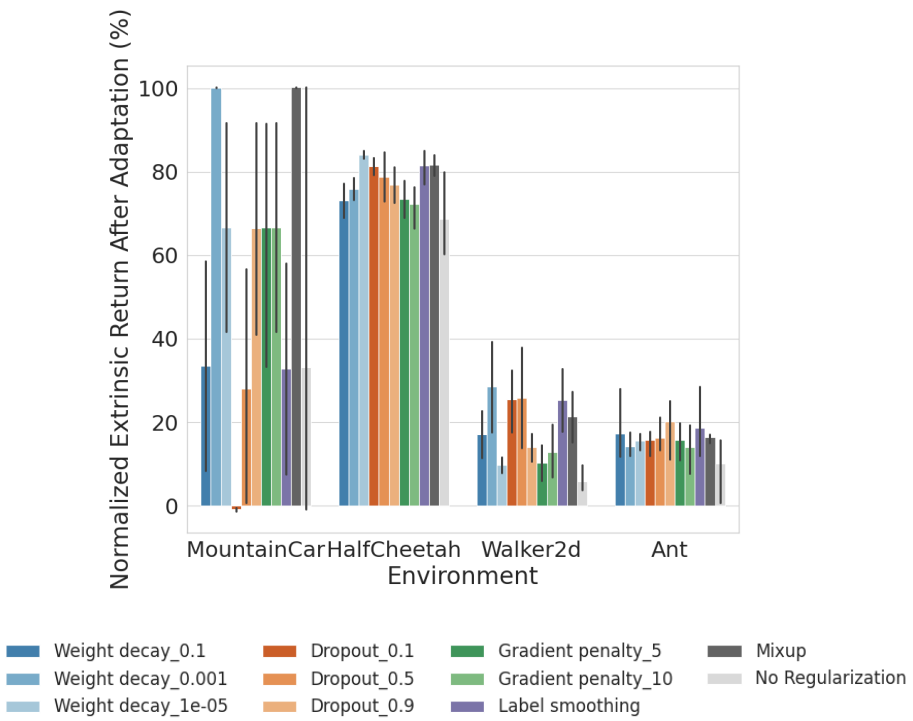

Figure 3: **Effect of regularization on downstream performance:** Regularization improved the results of downstream performance, weight decay and dropout gave the largest improvement.

4.5 MI Lower bound and Downstream Performance (Q5)

In this experiment we investigate the relation between the MI lower bound and downstream performance, specifically, we examined the NWJ lower bound I_{NWJ} Nguyen et al. (2010), InfoNCE I_{NCE} van den Oord et al. (2019), and α Choi et al. (2021), which are shown in equation 3, we used the separable architecture of the discriminator, in which we encoded the state and skill using an MLP encoders $\phi_1(s)$ and $\phi_2(z)$, we chose the score function as the dot product $f(s, z) = \phi_1(s)^\top \phi_2(z)$

The plots in figure 4, show the downstream performance for each lower bound.

I_{NCE} and I_{NWJ} consistently yielded higher downstream performance than the original I_{BA} (which corresponds to the original DIAYN implementation), I_α gave better returns than the I_{BA} in most cases, but it was worse than the I_{NCE} and I_{NWJ}.

This shows that the choice of the MI lower bound and the discriminator architecture is important for adaptation efficiency. Some lower bounds could give better performance than others.

4.6 What aspects of the environment do these representations learn? (Q6)

This experiment aims to show that the learned representations in the pre-training phase correspond to the controllable aspects of the environment.

We trained DIAYN with a separable discriminator and the InfoNCE lower bound on the Inverted-Pendulum environment. We added three random components to the state vector, each component was sampled from a standard Gaussian distribution $\mathcal{N}(\mu, \sigma)$. After convergence of the pre-trained policy, we trained a linear regression on the features from the state encoder. We report the mean square error (MSE) per state component in figure 5 to show which element of the state the linear regression could not fit well.

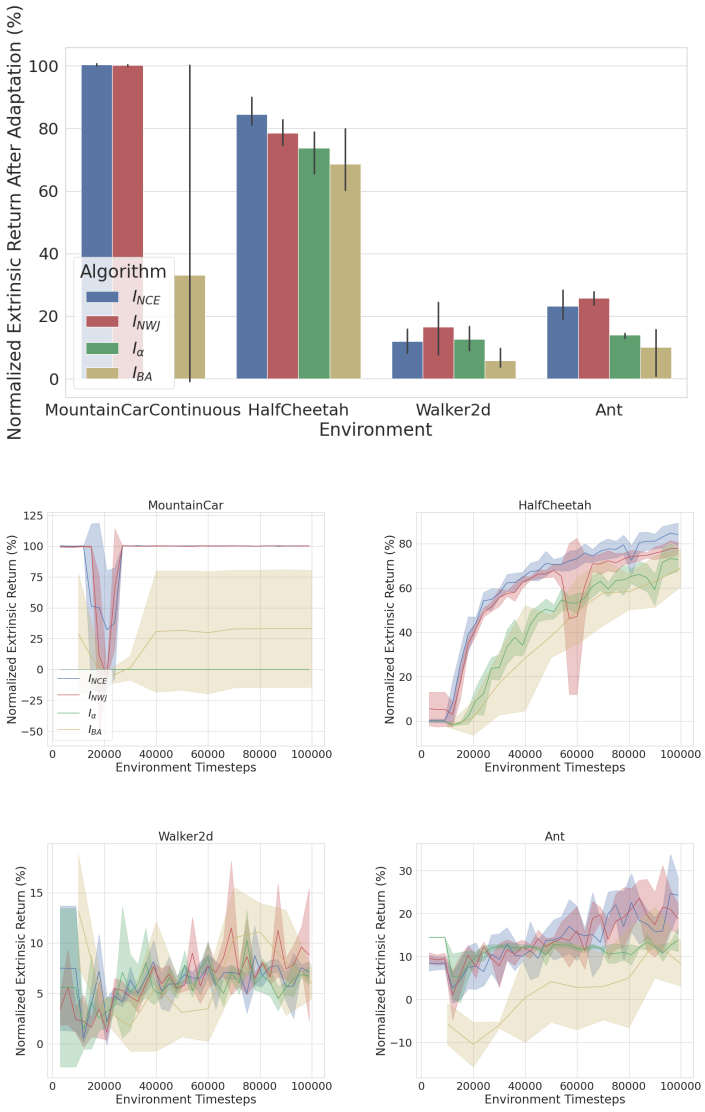

Figure 4: **MI lower bounds and downstream performance:** Both I_{NCE} and I_{NWJ} have better downstream performance than the original I_{BA}, I_{α} gave better returns than the I_{BA} in most cases, but it was worse than the I_{NCE} and I_{NW}.

We observe that the MSE is larger for components that are difficult to control. For example, it is easier to control the cart position than to control the cart velocity, and the MSE reached the same value for all uncontrollable noise components.

5 Conclusion

Unsupervised skill learning methods are a form of unsupervised pre-training for RL that has the potential to improve the sample efficiency of solving downstream tasks. Prior work has proposed several methods for unsupervised skill discovery based on MI objectives, with different methods varying in how this mutual information is estimated and optimized. This work showed some key design decisions could affect the sample efficiency of solving downstream tasks. Our key findings are that the sample efficiency of downstream adaptation under off-policy backbones is better than their on-policy counterparts. In contrast, on-policy backbones resulted in better state coverage,

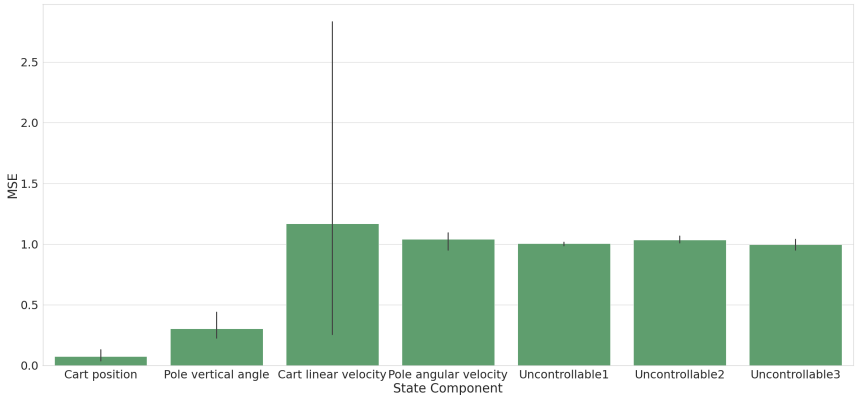

Figure 5: **MSE per state component:** The controllable components of the environment have lower MSE than the components that are difficult to control.

regularizing the discriminator gave better results, and careful choice of the mutual information lower bound and discriminator architecture yielded significant improvements on downstream tasks.

The limitations of our work are that we did not explore other optimization backbones such as black-box optimizers or model-based RL methods, and we did not examine other design choices such as the skill parametrization and the scalability with respect to the number of skills. We leave that for feature work.

Also, the questions of scalability, MI lower bound, and the discriminator architecture, open a lot of potential research to improve our understanding of these factors.

ACKNOWLEDGMENTS

We thank Abdoulaye Diack and Jonathan Caton from Google AI for providing us with google cloud platform (GPC) credit to run our experiments.

REFERENCES

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms, 2018.

David Barber and Felix V. Agakov. The im algorithm: A variational approach to information maximization. In *NIPS*, pp. 201–208, 2003. URL `http://papers.nips.cc/paper/2410-information-maximization-in-noisy-channels-a-variational-approach`.

Kate Baumli, David Warde-Farley, Steven Hansen, and Volodymyr Mnih. Relative variational intrinsic control, 2020.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giro i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Jongwook Choi, Archit Sharma, Honglak Lee, Sergey Levine, and Shixiang Shane Gu. Variational empowerment as representation learning for goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pp. 1953–1963. PMLR, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

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.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

Yifan Gao, Chien-Sheng Wu, Jingjing Li, Shafiq Joty, Steven C. H. Hoi, Caiming Xiong, Irwin King, and Michael R. Lyu. Discern: Discourse-aware entailment reasoning network for conversational machine reading, 2020.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018b.

Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding, 2020.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. CIC: Contrastive intrinsic control for unsupervised skill discovery. In *Deep RL Workshop NeurIPS 2021*, 2021a. URL https://openreview.net/forum?id=Z12zA99EFEi.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: Unsupervised reinforcement learning benchmark. In *Deep RL Workshop NeurIPS 2021*, 2021b. URL https://openreview.net/forum?id=OLAYpF9TQtQ.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching, 2020.

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.

Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features, 2021a.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *arXiv preprint arXiv:2103.04551*, 2021b.

XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, Nov 2010. ISSN 1557-9654. doi: 10.1109/tit.2010.2068870. URL http://dx.doi.org/10.1109/TIT.2010.2068870.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. *arXiv preprint arXiv:2102.11271*, 2021.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 7444–7453. PMLR, 2019.

Rui Zhao, Yang Gao, Pieter Abbeel, Volker Tresp, and Wei Xu. Mutual information state intrinsic control, 2021.

A Experimental Details

B regularization Experiment Results

Table 2 show the results for extrinsic rewards, state entropy, and intrinsic reward in the regularization experiments, and figure 6 show the learning curves of the adaptation phase.

C Hyperparameters

We ran the experiments in parallel on GCP, with 32 Intel Cascade Lake CPUs and 128 GB of RAM, the average time of the experiments was about 18 hours, tables 3 and 3 show the hyperparameters used in DIAYN, we used 30 skills for all environments except MountainCar where we used 10 skills.

Environment	Regularization	$\mathbf{R_{ext}}$ After Adaptation (%)	$\mathbf{R_{ext}}$ Before Adaptation (%) (Best Skill)	$\mathbf{R_{int}}$	State Entropy
Ant	Dropout_0.1	15.73 ± 3.26	-2.46 ± 2.81	3286.66 ± 113.59	-1130.67 ± 4.33
	Dropout_0.5	16.35 ± 4.29	1.23 ± 2.97	2822.25 ± 984.69	-1137.64 ± 17.42
	Dropout_0.9	20.17 ± 7.82	5.92 ± 0.30	3114.92 ± 129.75	-1148.46 ± 18.16
	Gradient penalty_10	13.97 ± 5.85	-1.63 ± 3.08	1983.96 ± 2217.11	-1128.94 ± 20.02
	Gradient penalty_5	15.80 ± 4.54	-1.62 ± 2.77	596.26 ± 2295.51	-1130.22 ± 18.51
	Label smoothing	18.68 ± 8.77	4.07 ± 0.72	2982.85 ± 109.98	-1134.82 ± 18.97
	Mixup	16.38 ± 1.21	0.75 ± 0.31	3291.15 ± 58.01	-1133.86 ± 7.83
	No Regularization	10.14 ± 8.14	0.78 ± 2.29	2392.13 ± 1700.34	-1134.03 ± 20.06
	Weight decay_0.001	14.22 ± 3.41	4.09 ± 1.24	3360.50 ± 19.71	-1131.64 ± 25.49
	Weight decay_0.1	17.25 ± 9.32	14.35 ± 0.10	0.61 ± 23.53	-1068.60 ± 20.79
	Weight decay_1e-05	15.50 ± 2.02	1.78 ± 0.84	3121.76 ± 152.71	-1126.40 ± 11.17
HalfCheetah	Dropout_0.1	81.27 ± 3.39	-2.21 ± 0.10	3397.77 ± 0.59	-166.31 ± 2.42
	Dropout_0.5	78.82 ± 9.28	-1.84 ± 0.14	3389.34 ± 12.28	-161.93 ± 9.32
	Dropout_0.9	76.85 ± 6.50	-1.53 ± 0.44	3226.64 ± 30.71	-169.73 ± 0.72
	Gradient penalty_10	72.18 ± 8.34	-0.61 ± 2.49	3261.09 ± 200.60	-166.69 ± 2.27
	Gradient penalty_5	73.48 ± 6.95	-0.59 ± 2.52	3262.69 ± 198.60	-166.71 ± 2.28
	Label smoothing	81.50 ± 6.58	-1.31 ± 0.15	3064.00 ± 5.45	-170.18 ± 0.21
	Mixup	81.62 ± 3.95	-2.35 ± 0.16	3373.02 ± 10.07	-169.50 ± 0.51
	No Regularization	68.61 ± 10.15	-1.73 ± 0.03	3252.04 ± 199.50	-156.27 ± 14.70
	Weight decay_0.001	75.87 ± 4.10	0.67 ± 3.73	3377.67 ± 3.86	-167.41 ± 2.61
	Weight decay_0.1	73.18 ± 6.98	0.18 ± 0.05	156.19 ± 241.84	-164.63 ± 1.76
	Weight decay_1e-05	84.06 ± 1.58	-1.71 ± 0.50	3396.05 ± 1.36	-167.78 ± 2.93
MountainCar	Dropout_0.1	-0.85 ± 0.84	-1.12 ± 1.04	2122.50 ± 219.25	-12.31 ± 1.55
	Dropout_0.5	28.10 ± 53.18	-2.07 ± 1.38	2040.78 ± 130.96	-9.05 ± 1.81
	Dropout_0.9	66.44 ± 49.85	64.29 ± 49.80	1680.72 ± 11.09	-11.46 ± 0.97
	Gradient penalty_10	66.68 ± 49.32	65.56 ± 48.70	2045.14 ± 174.20	-9.99 ± 0.84
	Gradient penalty_5	66.59 ± 49.23	65.58 ± 48.82	2109.38 ± 87.64	-9.99 ± 0.85
	Label smoothing	32.81 ± 49.73	32.94 ± 49.59	1864.00 ± 90.73	-10.75 ± 1.26
	Mixup	100.20 ± 0.18	63.33 ± 52.30	1851.42 ± 134.08	-10.31 ± 1.11
	No Regularization	33.11 ± 58.15	63.88 ± 61.89	1931.84 ± 352.14	-9.64 ± 1.39
	Weight decay_0.001	100.17 ± 0.11	99.44 ± 0.47	1550.80 pm 213.09	-10.86 ± 0.99
	Weight decay_0.1	33.44 ± 49.41	67.47 ± 49.84	-17.85 ± 7.04	-9.80 ± 1.14
	Weight decay_1e-05	66.63 ± 49.25	64.63 ± 50.57	1792.28 ± 472.14	-9.62 ± 0.82
Walker2d	Dropout_0.1	25.44 ± 11.05	12.14 ± 4.85	3065.32 ± 269.73	-7.32 ± 31.37
	Dropout_0.5	25.88 ± 16.45	16.31 ± 0.80	3219.79 ± 140.15	-40.90 ± 4.17
	Dropout_0.9	13.97 ± 4.41	10.19 ± 4.37	2628.64 ± 334.76	-10.43 ± 24.86
	Gradient penalty_10	12.89 ± 9.54	16.03 ± 0.90	1325.94 ± 2479.83	-14.02 ± 12.35
	Gradient penalty_5	10.24 ± 6.05	16.02 ± 0.90	1125.21 ± 2334.75	-15.59 ± 11.80
	Label smoothing	25.30 ± 11.57	5.34 ± 0.50	2955.23 ± 38.52	22.80 ± 5.88
	Mixup	21.31 ± 8.24	9.63 ± 7.19	3001.33 ± 283.27	16.03 ± 18.68
	No Regularization	5.84 ± 3.38	16.57 ± 1.7	3342.23 ± 64.15	-16.14 ± 12.7
	Weight decay_0.001	28.47 ± 15.05	17.18 ± 0.44	2903.20 ± 633.91	-27.36 ± 16.56
	Weight decay_0.1	17.18 ± 8.13	8.06 ± 1.72	428.71 ± 146.58	-20.83 ± 10.79
	Weight decay_1e-05	9.82 ± 2.54	12.69 ± 4.87	1849.87 ± 1573.23	-3.43 ± 32.23

Table 2: Results of Regularization Experiment

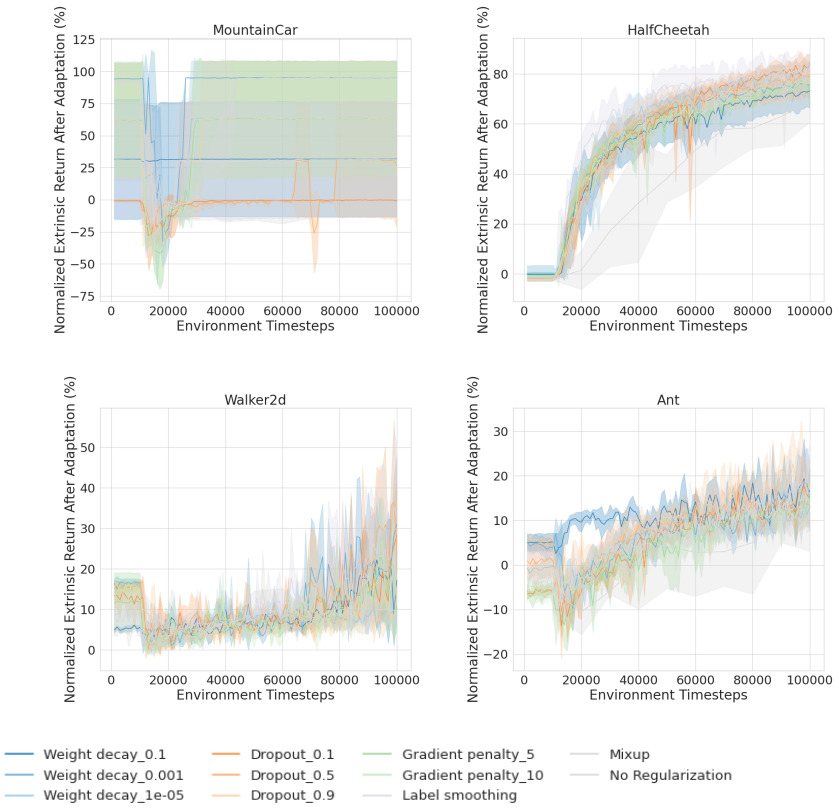

Figure 6: Learning Curves of Regularization Experiment

Table 3: DIAYN (SAC)

Hyperparameter	Value
Learning rate	3×10^{-4}
γ	0.99
Buffer size	10^7
Batch size	128
τ	0.001
Gradient steps	1
Entropy coefficient	0.1
Learning start	10^4

Table 4: DIAYN (PPO)

Hyperparameter	Value
Learning rate	$1 - 4$
γ	0.99
Batch size	4096
Rollout size	2048
clip	0.1
No. actors	32
λ	0.95
Entropy coefficient	0.1
Epochs	10

D Qualitative Analysis

In this section, we introduce a qualitative analysis of the learned skill in a point-mass environment. We show the effect of some of the design decisions on the skills learned by the original DIAYN implementation. Figure 7shows the skills distributions over the x,y position of the point-mass of SAC vs PPO. Figure 8 shows the skills distributions resulting from applying weight decay and mixup to DIAYN with SAC.

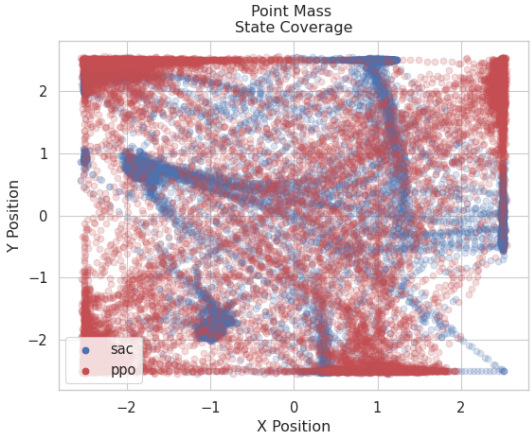

Figure 7: **Skills distributions over the x,y position of the point-mass of SAC vs PPO:** The plot shows the visited states by each algorithm. This plot supports the results in table 1, in which PPO had higher state entropy in most environments while SAC had more specialized(in this plot, more clustered) skills.

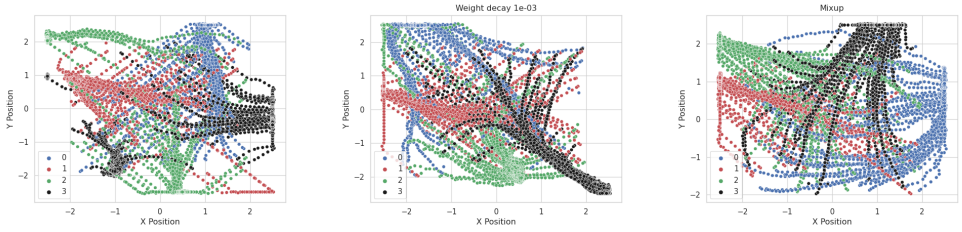

Figure 8: **Skill distribution under different regularizers:** Without regularization (the left figure), the learned skills are clearly separable, and some skills visit two separate regions of the state space, adding regularization improves the discriminability of the learned skills and guide each skill to focus on a specific region of the state space.