SCALING MUTUAL INFORMATION SKILL LEARNING TO LARGER SKILL SPACES

Devan Shah & David Shustin
Department of Computer Science, Princeton
{devan.shah,dshustin}@princeton.edu

1 INTRODUCTION

In many domains of machine learning, high-dimensional embeddings are important for expressive representation learning. As deep reinforcement approaches have reached and even surpassed human performance in many domains, such as playing Atari games (17), designing hardware circuits (25), solving Rubik's cubes (29), and dominating collaborative-competitive games like Monopoly (4) and Diplomacy (2), we would expect that increasing behavioral complexity requires higher-dimensional embeddings of behavior, as they have in other representation learning tasks in image generation (14) and image-language learning (21). Much of the success in the aforementioned tasks using conventional RL can be attributed to improved algorithms that better explore the possible states and to better models of the environment. These algorithms typically allow the agent to attempt different actions and receive a reward corresponding to furthering along the desired goal (i.e., solving a side of the Rubik's cube). Traditional deep reinforcement learning approaches often rely on a model of the environment as a Markov Decision Process, where there are states $s_t \in \mathcal{S}$ and, after taking an action $a_t \in \mathcal{A}$, the next state s_{t+1} is sampled from the distribution $p(s|s_t, a_t)$, where p corresponds to the environment transition probabilities. As part of the model, we design rewards $r(s_t, a_t)$ for taking actions at particular states. The reinforcement learning problem is thus to learn a policy $\pi(a|s_t)$, which is a distribution over actions from a given state, that maximizes the expected cumulative discounted reward from the environment, e.g.

$$\pi^* = \text{argmax}_\pi \mathbb{E}_{s_{t+1}\sim p(\cdot|s_t,a_t), a_t\sim\pi(\cdot|s_t)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$$

However, we may have access to an environment but lack reward signals, motivating an *unsupervised* approach to reinforcement learning. In many real-world domains, it can be challenging to shape a reward signal resulting in the desired behavior (8) or to gather labeled trajectories, and additionally we may wish to design agents with policies that can be easily adapted to many different rewards. This challenge motivates the development of unsupervised reinforcement learning, in which the agent has access to an environment but primarily explores the environment in the absence of a reward signal (8; 12). In this setting, the agent often navigates the environment without rewards (unsupervised period), followed by a substantially shorter exploration phase with rewards (supervised period) where the agent learns to adapt its knowledge to the new observed reward signal (12). We discuss specific approaches for unsupervised reinforcement learning in Section 2.

One approach we will focus on for reinforcement learning involves training the reinforcement learning agent without rewards before specializing it on a reward signal. We use unsupervised reinforcement learning to teach the agent a diverse set of "skills" conditioning the policy's behavior during the unsupervised training period, and then learn to adapt those skills to collect reward during a supervised period (12). Mutual Information Skill Learning (MISL) attempts to learn, for each latent skill $z \sim Z$, a distinct trajectory across the environment. More specifically, for a fixed skill distribution Z, we train a skill-conditioned policy π optimizing:

$$\mathcal{I}_{z\sim Z,\tau\sim\pi(\cdot|z)}(f(\tau); z) := \mathcal{H}(f(\tau)) - \mathcal{H}(f(\tau)|z) \tag{1}$$

where τ corresponds to a trajectory rollout $(s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$ from policy π, where we sample $a_t \sim \pi(s_t, z)$ and $s_{t+1} \sim p(s|s_t, a_t)$. f is often either the identity function or returns the

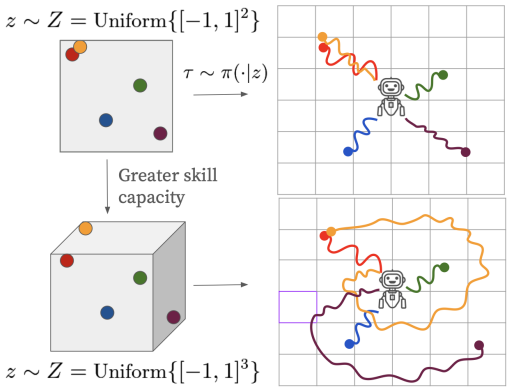

Figure 1: With a higher dimensional skill latent, MISL-like methods should have the capacity to learn more aspects of the state space and a greater diversity of trajectories. At minimum, high-dimensional MISL should explore at least as well as with a lower dimensional state space. This is an illustrative figure.

final state, depending on whether MISL aims to learn skills corresponding to distinct trajectories or distinct final states (8; 16; 1). We can decouple the Mutual Information objective into the terms $\mathcal{H}(\tau)$, which incentivizes the model to learn a policy that covers a large region of the trajectory space, and into the term $-\mathcal{H}(\tau|z)$, which incentivizes the resulting trajectory to be deterministic given z. Thus, ideally, the policy learns distinct skills that correspond to executing distinct trajectories and reaching distinct regions of the state space.

Papers such as VISR (12) show that models trained with unsupervised RL learn expressive behaviors, which can later be adapted to specific reward signals. For increasingly complex and large environments, successful unsupervised training with Mutual Information Skill Learning will require learning a large class of skills in order to parametrize a diverse class of downstream behaviors, and we expect that complex behavior is best expressed in a high-dimension skill space, as illustrated in Figure 1.

However, in prior papers leveraging Mutual Information Skill Learning, skills are often drawn from small categorical distributions or low-dimensional subspaces. VISR (12) tests skill dimensionality ranging from $d = 2$ to $d = 50$ and observes the best performance by choosing skills from a subset of $\mathbb{R}^5$. DIAYN (8) chooses skills from a categorical distribution with $K = 50$ entries, DISCS (13) chooses skills from a subset of $\mathbb{R}^2$ (16), and CSF (28) performs dramatically worse with latents in $\mathbb{R}^8$ or $\mathbb{R}^{32}$ compared to $\mathbb{R}^2$. Thus current MISL approaches struggle to adapt to high-dimensional skill latents. **To extend MISL to handle more complex environments, it is critical that MISL approaches can be adapted to effectively learn distinct skills from high-dimensional skill spaces.**

We explore competing hypotheses on why MISL fails to produce effective skills for downstream tasks when skills are sampled from a high-dimensional subspace, with a goal of better understanding the challenges in this domain and to develop training methods mitigating this concern.

2 Related Work

In this section, we discuss the relevant prior work regarding unsupervised reinforcement learning, unsupervised skill learning, and the current limitation and research within mutual information skill learning.

Unsupervised Reinforcement Learning. Many techniques in unsupervised reinforcement learning (7; 3; 12; 9) involve assuming $r(s_t, a_t) \approx \phi(s_t, a_t, g)^\intercal g$, for some goal vector g and a mapping from states to state features ϕ. With this substitution, we can represent the cumulative discounted reward in terms of the feature encoding:

$$\mathbb{E}\left[\sum_{t=0}^{T}\gamma^t r(s_t,a_t)\right] = \mathbb{E}\left[\sum_{t=0}^{T}\gamma^t \phi(s_t,a_t,g)^\top g\right] = \mathbb{E}\left[\left(\sum_{t=0}^{T}\gamma^t \phi(s_t,a_t,g)\right)^\top g\right]$$
$$= \mathbb{E}\left[\psi(s_0,a_0,g)^\top g\right]$$

where we define $\psi(s_0,a_0,g) = \sum_{t=0}^{T}\gamma^t\phi(s_t,a_t,g)$. During the unsupervised training period, as we do not know the reward, we sample random vectors g, and train a neural-network approximation to learn $\psi(s_t,a_t,g)$ based on model trajectories under an ϵ-greedy policy $\pi(s_t) = \text{argmax}_a \psi(s_t,a,g)^\top g$. During the supervised training period, we fix the encoder ψ and restrict ourselves to finding g that best models the shown rewards, often by linear regression (3).

This strategy summarizes a multitude of approaches including (3; 7; 9), with notable distinctions being that DFP (7) chooses ϕ corresponding to selecting the subset of state that matters for a reward, and successor features (3) uses a fixed feature encoder ϕ. CRL (9) learns the encoder ψ directly and a separate goal-state encoder ψ_g with NCE-binary loss, so that $\exp(\sigma(\psi(s,a)^\top\psi_g(s_g)))$ corresponds to the cumulative discounted reward of $r_g(s_t,a_t) \propto p(s_{t+1}=s_g|s_t,a_t)$. Other methods (12; 28) learn successor features in an unsupervised skill-learning manner described in the next section. Generally, the approaches we have shown decouple unsupervised reinforcement learning into predicting future states to learn ψ, which simply requires trajectory data rather than rewards. Many other unsupervised reinforcement learning techniques (6) can be viewed as predicting functions of future states for states gathering during an unsupervised training period or offline training period.

Many techniques in unsupervised reinforcement learning involve assuming a distribution of rewards or goals the agent cares to learn during the long unsupervised training period and predict future state evolution, either as rewards, cumulants over future states, or as distributions of future states, as a function of these goals (7; 3). Thus, for the supervised training period, the agent can find the goal in its distribution that best matches the observed rewards, and leverage the policy learned during unsupervised training period.

Unsupervised Skill Learning. Our work explores properties of skill-learning algorithms. In many ways, the above unsupervised reinforcement learning framework can also be thought of describing a skill-learning algorithm, where each vector g corresponds to a different "skill" conditioning the models behavior. Mutual Information Skill Learning methods are distinct in that they do not consider the representation of the reward or eventual goal and instead arise out of the intuition of allowing our agent to exert maximal control the environment (18), beyond just reaching high-reward regions, and leveraging the skill distribution to parameterize this control.

Note that the mutual information objective described in Eq. 1 is equivalent to:

$$\mathcal{I}(f(\tau);z) := \mathcal{H}(z) - \mathcal{H}(z|f(\tau)) \tag{2}$$

The mutual information objective described above is more commonly considered since, as z is drawn from a fixed distribution, $\mathcal{H}(z)$ is fixed and thus we solely optimize against $\mathcal{H}(z|f(\tau))$. Unfortunately, this objective is intractable, and thus prior work often considers the variational lower bound of $\mathbb{E}[q(z|f(\tau))]$ where q is a learned discriminator (12).

Prior work on Mutual Information Skill Learning. DIAYN (8) introduces unsupervised mutual information skill learning, leveraging the variational bound considered above, and considers the mutual information between the final state of a trajectory and the latent state. In navigation environments, DIAYN showcases skills corresponding to reaching diverse regions of the environment, and in locomotion tasks, skills corresponding to walking, running, hopping, etc. (8).

Subsequent work VALOR (1) improves skill learning capacity by training the discriminator q as an LSTM (24) network with input τ and introduces curriculum skill learning. They find the more expressive decoder allows for more distinct skills and train policies with hundreds of distinct categorical skills. DADS (23) replaces the discriminator $q(z|s)$ with a dynamics model $q(s'|z,s)$, thus aiming to associate experienced trajectories to particular skills. CIC (16) further tests the hypothesis that the discriminator capacity limits skill learning, and designs a contrastive discriminator learned with the NCE objective (10) that better scales to large skill vectors. CSF (28) also uses contrastive learning to learn an actor but uses a successor feature-like approach to learn the critic. VISR (12)

merges the MISL objective with the unsupervised reinforcement learning $r_g(s,a) = \phi(s)^\top g$ and trains successor features $\psi(s,a,g)$. However, VISR optimizes r_g rather then a cumulant, leading to a interpretation of z as both goals and direction in state-space that VISR wishes to optimize alignment with.

Performance of High-dimensional Skill Learning. Many unsupervised skill learning methods use a low-dimensional space for embedding skill vectors. In addition, the related works report that unsupervised skill learning typically behaves worse as latent dimension is increased. DIAYN leverages skills drawn from a uniform distribution over a size 50 categorical distributions, however, subsequent work (12) finds improved downstream performance with skills drawn from a the uniform distribution on the 4-sphere, $\{x \in \mathbb{R}^5 : ||x||_2 = 1\}$. DADS (23) only leverages 20 discrete skills for some tasks, and $2-5$ dimensional continuous distributions for other tasks. VISR (12) sweeps over embedding dimensions for the hypersphere from 2 to 50 and determines that 5 is the best performing dimensionality. DISCS (13) is a modification of VISR using a soft actor-critic (11) objective, which uses 2-dimensional space and reports worse performance in 3 or 4 dimensions. CSF (28) reports much worse performance with 8 and 32 dimensions than with 2 dimensions, as shown in Figure 2.

Enabling High-dimensional Skill learning. Several prior works have also expressed interest in limitations on the skill dimension. Some of these works propose methods to mitigate this limitation and enable a more complex skill distribution. For example, VALOR (1) proposes a curriculum learning approach in which the skill dimension is gradually increased during the course of training. Specifically, if K is the number of available skills, a skill is drawn according to $c \sim \text{Categorical}(K)$. The policy π is parametrized by the drawn skill c, and a trajectory is sampled according to the policy and the environment dynamics $\tau \sim \pi(c)$. K is updated according to an exponential schedule (1)

$$K \leftarrow \min(\lfloor 1.5K + 1 \rfloor, K_{\max})$$

when the performance of the decoder $\mathbb{E}_{\substack{z\sim\text{Cat}(K) \\ \tau\sim\pi(\cdot|z)}}[p(z|\tau)]$ reaches a fixed threshold, meaning that the decoder is powerful enough to learn more skills. CIC (16) is similarly concerned by the inability of MISL methods to adapt to high-dimensional skill latents, and offers an improved discriminator as a potential solution.

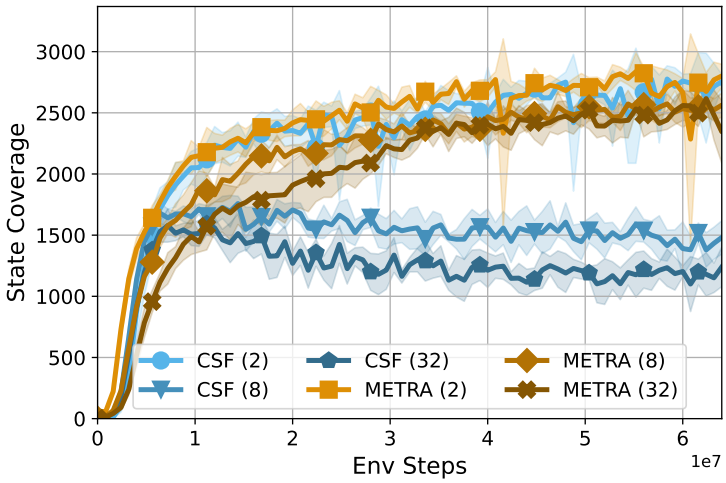

Figure 2: Performance of CSF (28) and related method METRA (20) against a sweep of skill dimension. This is Figure 9 in (28).

3 METHODS

In this work, we propose several hypotheses for why high-dimensional skill vectors are difficult to learn in the MISL setting. For each hypothesis, we will evaluate experiments that can support or falsify the hypothesis.

We perform experiments using VISR (12), a popular mutual information skill learning approach. For VISR, the skill performance is measured by the reward on a downstream task after generalized policy improvement with the skills. We evaluate VISR on the MinAtar Atari-inspired environments of Breakout, Space Invaders, and Asterix (27) as implemented in `gymnax` (15). In our Appendix, we additionally include preliminary experiments with CSF (28), a more recent MISL method. Implementation details for VISR and CSF are available in the Appendix.

We choose to evaluate VISR because its admits a simple method to apply learned skills to inference-time tasks by leverage linear regression to find a skill most similar to reward-maximizing behavior. This allows us to evaluate our models by measuring the average reward. There are also efficient VISR implementations using JAX that we use to run our experiments (22). Both CSF and VISR are known to perform worse when the skill dimension is higher, as demonstrated for CSF in Figure 2 and mentioned in the VISR skill-dimension sweep (12). In the main paper, we will predominantly consider the following with VISR:

Baselines. Before we can ablate the model and understand the limitations of high-dimensional skill vectors, we must establish the baseline performance of VISR in our chosen environments. We will perform a sweep over latent dimension d to determine the best-performing value for d and confirm that performance degrades as d increases.

Encoder expressivity. We will investigate improving the expressivity of the VISR ψ encoder. We hypothesize that Markovian encoders, which depend on the current state, action, and skill of the trajectory have more limited capacity to predict state evolution than if they had the prior trajectory. Even in cases such as Atari games, we predict past temporal information can aid in predicting future skill-conditioned state evolution.

Environment Complexity. We expect that more complex environments have more skills to learn, and thus are more suitable for high skill latent dimension d. As a proxy for this, we additionally expect environments with longer horizon to be more suitable for larger d.

Skill localization. In high dimensions, VISR may struggle more with policy improvement, as we have more axes to optimize over, and thus a sampled skill is less likely to be relevant. Despite having relevant skills, the increased state-space freedom may demand more samples for VISR generalized policy improvement to learn useful ψ. Thus, one hypothesis is that VISR with high d is starved for samples for policy improvement, and will have disproportionate improvement with more generalized policy improvement samples.

Training time. It is possible that existing approaches for skill learning are already capable of learning skills from subsets of high-dimensional $\mathbb{R}^d$, but since each skill has fewer trajectories, we simply require scaling training time proportionally to the amount of trajectories.

4 EVALUATIONS

In this section, we will formalize our hypotheses and perform experiments to test their validity.

We hypothesize that increased training time, better skill localization, longer environments, and non-Markovian discriminators will lead to disproportionate improvements on downstream tasks for VISR with larger skill dimension ($d \geq 10$) relative to smaller skill dimension ($d < 10$).

To evaluate the hypothesis, we perform the following experiments:

1. We establish clear baselines to confirm the VISR (12) result that increasing skill dimensions leads to worse downstream performance.
2. We evaluate whether leveraging more skills for GPI leads to improved relative performance for higher skill-dimension models.
3. We evaluate whether increased environment complexity, in the form of longer trajectories, leads to improved performance for higher skill-dimension models.
4. We evaluate whether very long training runs allows for better learning in the form of downstream performance.

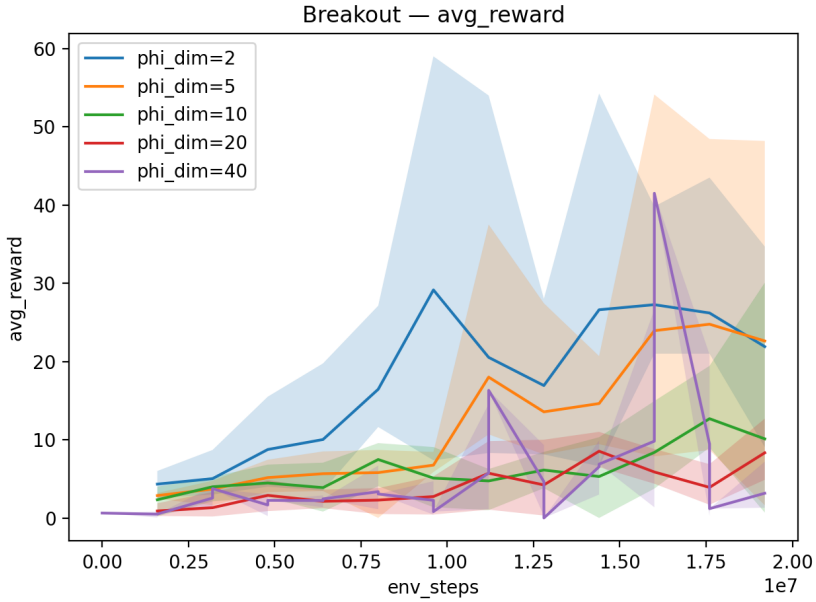

Figure 3: Average reward on the Breakout MinAtar (27) Game for a VISR model trained with varying skill dimension d and default parameters (see Appendix) The dark line is the mean of three runs, with the shaded regions showing the max and min of trials. For more details on model architecture and training, see Appendix. As a reference point, a DQN on Breakout achieves a reward of approximately 10 by 5×10^6 training frames (27), which is comparable to the $d = 2$ reward.

5. We ablate expressiveness and temporal-dependency for ψ, testing CNN-LSTMs of varying hidden-dimension and CNN models, to test whether more expressive discriminators improved higher skill dimension models.

Training Setup. We train VISR models with a convolutional backbone on an LSTM with hidden dimension 128. The encoder ϕ leverages a CNN architecture while ψ leverages a CNN-LSTM architecture. Inference is performed by determining w_{inferred} such that $\phi(s)^\top w_{\text{inferred}} \approx r_s$ by least-squares, and taking the action maximizing $\psi(s, a, w_{\text{inferred}})^\top w_{\text{inferred}}$ at each step. During training rollouts, we leverage generalized policy improvement (GPI), selecting $\texttt{num\_gpi\_samples} = 10$ distinct skills and choosing $a = \text{argmax}_{w \in W \cup \{z\}, a \in \mathcal{A}} \psi(s, a, w)^\top z$, where $|W| = \texttt{num\_gpi\_samples}$ and is sampled from the skill distribution, and z is the skill for that rollout. Note this GPI implementation differs from the traditional VISR (12) implementation as it does not sample from a Von-Mises Fisher distribution centered at a base skill. Full training details, default parameters, and model architectures are included in the Appendix.

For our codebase, we build on a VISR reimplementation: https://github.com/mjsargent/JAX_VISR (22). The Atari games used are the MinAtar (27) versions as implemented in `gymnax` (15). We additionally include preliminary CSF (28) experiments in the Appendix.

4.1 Baselines

We will now evaluate the baselines for VISR across the Breakout, Space Invaders, and Asterix Atari environments (15). We performed 3 different training runs and plot the mean, min, and max rewards of the runs at evaluation. We note in Figure 3 that lower skill-dimension (phi-dim in the graph) generally corresponds to increased rewards. Similarly, in Figure 4, we observe that in the SpaceInvaders environment, the lower skill-dimension models tend to do better, and this behavior is exceptionally apparent in the Asterix environment. Across the benchmarks, we find that $d = 2$ is optimal (with results differing slightly from (12), which use $d = 2$), and that increasing d often corresponds to decreased performance.

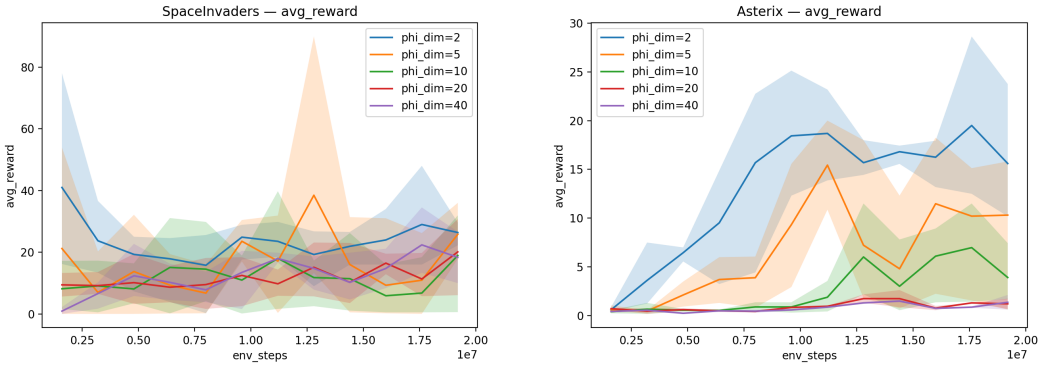

Figure 4: Average rewards on the Space Invader and Asterix MinAtar (27) Games for a VISR model trained with varying skill dimension d and default parameters (see Appendix) The dark line is the mean of three runs, with the shaded regions showing the max and min of trials. As reference points, a DQN on Space Invaders achieves a reward of approximately 50 by 5×10^6 training frames and a DQN on Asterix achieves a reward of approximately 17 by 5×10^6 training frames (27).

4.2 Generalized Policy Improvement evaluations

We study the effect of increasing the number of skills sampled during Generalized Policy Improvement (GPI) and present the results in Figure 5. Our implementation of VISR (22) samples uniformly over the skill space in addition to the rollout skill for choosing actions. This should allow for faster policy improvement, and mitigate the concern that in higher-dimensional spaces, generalized policy improvement requires more samples, as each sampled skill is likely less relevant to the rollout skill. However, we increased `num_gpi_samples` to 40 and 80 in the Breakout environment but did not observe an improvement in model performance for high-dimensional skills.

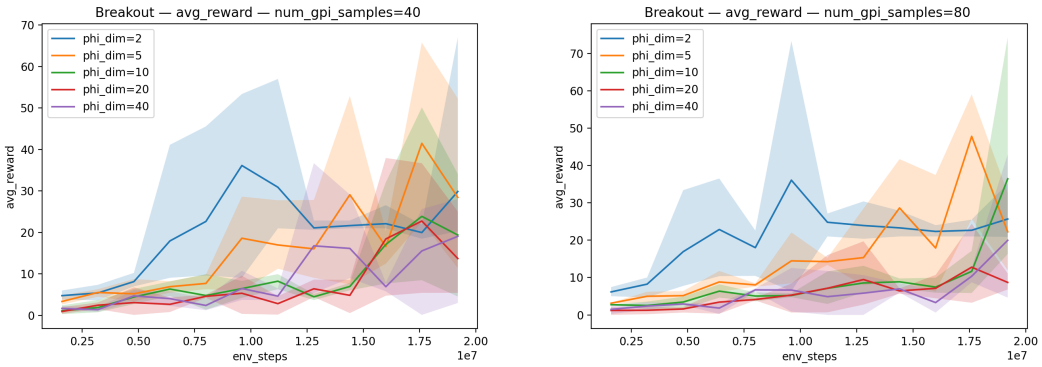

Figure 5: Average rewards on the Breakout MinAtar (27) Games for a VISR model trained with varying skill dimension d, ablated num_gpi_samples, and otherwise default parameters (see Appendix). We test alternate settings of num_gpi_samples $= 40$ and $= 80$, with the default of 10 used in Figure 3. The parameter num_gpi_samples represents the amount of samples used for generalized policy improvement.

4.3 Trajectory evaluations

We model the effect of increasing model complexity by increasing the trajectory length to 20 and 80 and present the results in Figure 6. Another datapoint is available in Figure 3 which uses the baseline trajectory length 40. Although the performance for $d = 10$ and $d = 40$ improve for higher trajectory length, the low-dimensional baselines $d = 2$ and $d = 5$ remain substantially stronger

than the high-dimensional ablations. Our results suggest that longer trajectories do not address the limitations of high-dimensional skills in VISR.

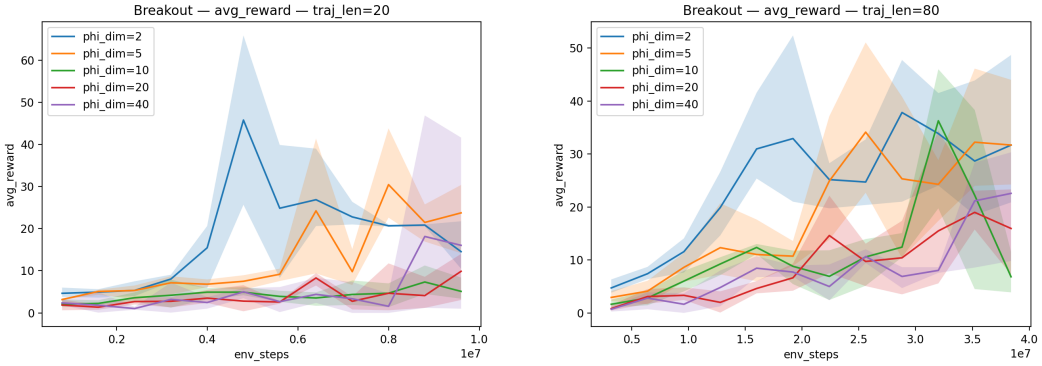

Figure 6: Average rewards on the Breakout MinAtar (27) Games for a VISR model trained with varying skill dimension d, ablated traj_len, and otherwise default parameters (see Appendix). We test alternate settings of traj_len $= 20$ and traj_len $= 80$, with the default of 40 used in Figure 3. The parameter traj_len represents the rollout trajectory length.

4.4 Length Evaluations

We trained our VISR network for $10\times$ as many steps as the base environment to determine whether higher skill dimensions are advantaged on a long timescale and present the results in Figure 7. Although the models tended to improve throughout training, there remained a gap between models trained with different skill dimensions. Models trained with dimension $d = 40$ and $d = 10$ underperformed the base model trained with $d = 2$.

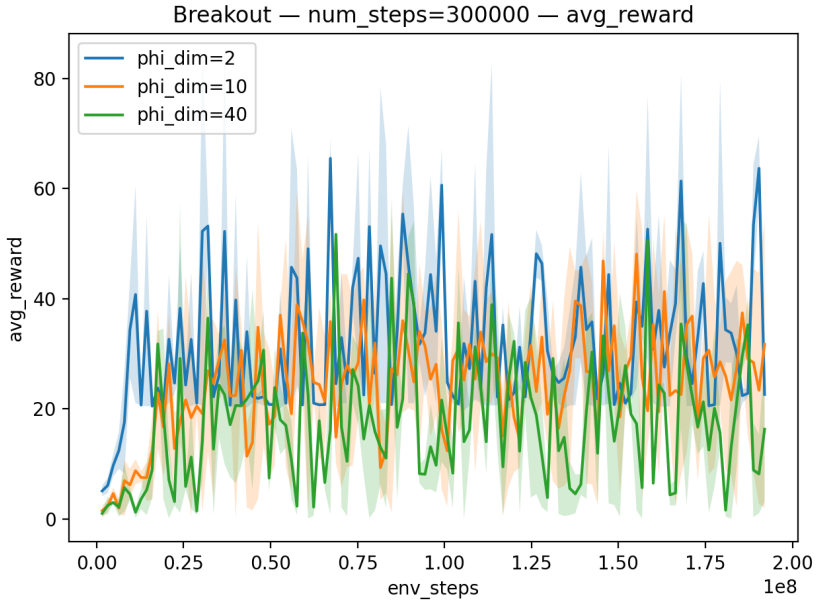

Figure 7: Average rewards on the Breakout MinAtar (27) Games for a VISR model trained with ablated training length and otherwise default parameters (see Appendix). We train with $\times 10$ as many training steps, with 2×10^8 total environment frames. We do not find that increased training time leads to disproportional improvement for larger d.

4.5 Model Capacity

We hypothesized that LSTM-based models, which can leverage the trajectory history, are better capable of learning interesting skills that span time. For this experiment, we consider two different model bases for ψ, we test LSTMs with hidden state $8, 32$, and 128, and we test a CNN with 16 features, 3×3 convolution region, and a downstream MLP with hidden dimension 128.

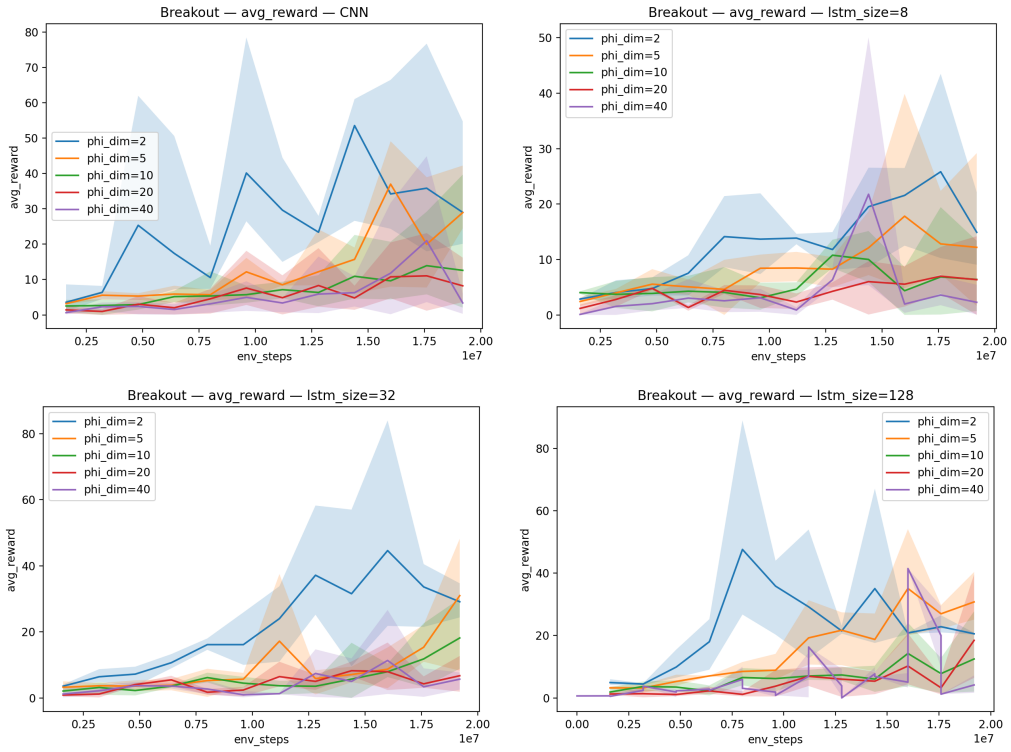

Figure 8: Average rewards on the Breakout MinAtar (27) Games for a VISR model trained with varying encoding architecture for ψ, varying skill dimension d, and otherwise default parameters (see Appendix). We test a CNN-MLP architecture and CNN-LSTM architecture for ψ, ranging the hidden dimension of the LSTM from 8 to 128 (baseline). We do not not notice any convincing trends between the encoder strength and performance for larger d.

5 Discussion and Conclusion

As confirmed in our experiments, MISL-based methods such as VISR tend to decrease in performance as skill dimension d increases. We hypothesized that increasing the length of training time, increasing trajectory length, and leveraging time-dependent encoders would lead to disproportionally stronger performance with high skill dimension. However, our experimental results do not show a convincing connection between any of the expected attributes and a disproportionate increase, leading us to reject our current hypotheses. We remain uncertain on practical techniques for improving performance with large d and whether this is crucial for more complex environment.

We have promising preliminary results for CSF in the Appendix, showing that larger hidden dimension corresponds to increased state-coverage on a block environment. However, due to only having the capacity for a single training run for each configuration, we require more experimentation before drawing conclusions in the CSF environment.

For future work, we are interested in continuing the CSF experiments and evaluating environments with more axis of movement and greater complexity.

6 ACKNOWLEDGEMENTS

We thank Chongyi Zheng and Catherine Ji for providing a baseline JAX implementation of the CSF method and their help in debugging experiments. We also thank Evan Dogariu and several members of our COS 597R seminar for their helpful feedback.

7 LLM ACKNOWLEDGEMENTS

LLMs were used as tools for programming and aided in code generation for figure creation, data parsing, debugging environments and models, setting up programming environment configurations, understanding new codebases, debugging latex, and implementing additional CSF evaluations (unused in the paper). LLMs were *not* used for experiment design, planning, writing, methodology, or anything that falls outside of aiding coding or aiding learning.

REFERENCES

[1] ACHIAM, J., EDWARDS, H., AMODEI, D., AND ABBEEL, P. Variational option discovery algorithms, 2018.

[2] BAKHTIN, A., WU, D. J., LERER, A., GRAY, J., JACOB, A. P., FARINA, G., MILLER, A. H., AND BROWN, N. Mastering the game of no-press diplomacy via human-regularized reinforcement learning and planning, 2022.

[3] BARRETO, A., DABNEY, W., MUNOS, R., HUNT, J. J., SCHAUL, T., VAN HASSELT, H., AND SILVER, D. Successor features for transfer in reinforcement learning, 2018.

[4] BONJOUR, T., HALIEM, M., ALSALEM, A., THOMAS, S., LI, H., AGGARWAL, V., KEJRIWAL, M., AND BHARGAVA, B. Decision making in monopoly using a hybrid deep reinforcement learning approach, 2022.

[5] BORTKIEWICZ, M., PAŁUCKI, W., OSTASZEWSKI, M., AND EYSENBACH, B. Is temporal difference learning the gold standard for stitching in rl?, 2025.

[6] CHEN, L., LU, K., RAJESWARAN, A., LEE, K., GROVER, A., LASKIN, M., ABBEEL, P., SRINIVAS, A., AND MORDATCH, I. Decision transformer: Reinforcement learning via sequence modeling, 2021.

[7] DOSOVITSKIY, A., AND KOLTUN, V. Learning to act by predicting the future, 2017.

[8] EYSENBACH, B., GUPTA, A., IBARZ, J., AND LEVINE, S. Diversity is all you need: Learning skills without a reward function, 2018.

[9] EYSENBACH, B., ZHANG, T., SALAKHUTDINOV, R., AND LEVINE, S. Contrastive learning as goal-conditioned reinforcement learning, 2023.

[10] GUTMANN, M., AND HYVÄRINEN, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010), Y. W. Teh and M. Titterington, Eds., vol. 9 of *Proceedings of Machine Learning Research*, PMLR, pp. 297–304.

[11] HAARNOJA, T., ZHOU, A., ABBEEL, P., AND LEVINE, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[12] HANSEN, S., DABNEY, W., BARRETO, A., DE WIELE, T. V., WARDE-FARLEY, D., AND MNIH, V. Fast task inference with variational intrinsic successor features, 2020.

[13] IMAGAWA, T., HIRAOKA, T., AND TSURUOKA, Y. Unsupervised discovery of continuous skills on a sphere, 2023.

[14] KARRAS, T., LAINE, S., AND AILA, T. A style-based generator architecture for generative adversarial networks, 2019.

[15] LANGE, R. T. gymnax: A JAX-based reinforcement learning environment library, 2022.

[16] LASKIN, M., LIU, H., PENG, X. B., YARATS, D., RAJESWARAN, A., AND ABBEEL, P. Cic: Contrastive intrinsic control for unsupervised skill discovery, 2022.

[17] MNIH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLOU, I., WIERSTRA, D., AND RIEDMILLER, M. Playing atari with deep reinforcement learning, 2013.

[18] MOHAMED, S., AND REZENDE, D. J. Variational information maximisation for intrinsically motivated reinforcement learning, 2015.

[19] PARK, S., FRANS, K., EYSENBACH, B., AND LEVINE, S. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)* (2025).

[20] PARK, S., RYBKIN, O., AND LEVINE, S. Metra: Scalable unsupervised rl with metric-aware abstraction, 2024.

[21] RADFORD, A., KIM, J. W., HALLACY, C., RAMESH, A., GOH, G., AGARWAL, S., SASTRY, G., ASKELL, A., MISHKIN, P., CLARK, J., KRUEGER, G., AND SUTSKEVER, I. Learning transferable visual models from natural language supervision, 2021.

[22] SARGENT, M. JAX_VISR: Jax re-implementation of variational intrinsic successor features. `https://github.com/mjsargent/JAX_VISR`, 2022. GitHub repository.

[23] SHARMA, A., GU, S., LEVINE, S., KUMAR, V., AND HAUSMAN, K. Dynamics-aware unsupervised discovery of skills, 2020.

[24] VENNERØD, C. B., KJÆRRAN, A., AND BUGGE, E. S. Long short-term memory rnn, 2021.

[25] VIJAYARAGHAVAN, P., SHI, L., DEGAN, E., MUKHERJEE, V., AND ZHANG, X. Autocircuit-rl: Reinforcement learning-driven llm for automated circuit topology generation, 2025.

[26] WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning 8*, 3 (May 1992), 229–256.

[27] YOUNG, K., AND TIAN, T. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments, 2019.

[28] ZHENG, C., TUYLS, J., PENG, J., AND EYSENBACH, B. Can a misl fly? analysis and ingredients for mutual information skill learning, 2025.

[29] ZIARKO, A., BORTKIEWICZ, M., ZAWALSKI, M., EYSENBACH, B., AND MILOS, P. Contrastive representations for temporal reasoning, 2025.

APPENDIX

VISR IMPLEMENTATION DETAILS

VISR Configuration		
	Description	**Default**
`num_gpi_samples`	Number of samples for GPI	10
`trajectory_length`	Rollout length	40
`num_eval_envs`	Evaluation Batch Size	16
`gamma`	Decay relationship between ψ and ϕ	0.95
`epsilon`	Policy exploration parameter	0.1
`lr`	Model learning rate	3×10^{-4}
`num_steps`	Amount of epochs	30,000
`target_update_frequency`	Updated frequency for target model	100
`recurrent`	Recurrent layers in model, 0 for no recurrence	1
`lstm_size`	Hidden dimension of LSTM or CNN model	128
`seed`	Randomness for training run. We always test with seeds 1, 2, 3.	[1,2,3]

Table 1: Default training configuration details for the VISR model runs.

The following is the CNN-LSTM default architecture used for Ψ in VISR (22).

```
class CNNLSTMPsiMultiBranch(nn.Module):

    phi_dim: int
    num_a: int
    lstm_size: int

    @nn.compact
    def __call__(self, x, w, lstm_state):
        # expects a tuple of (x, (h, c))
        x = nn.Conv(features=16, kernel_size=(3, 3))(x)
        x = nn.relu(x)
        x = x.reshape((x.shape[0], -1))  # flatten
        x = jnp.concatenate((x, w), axis=-1)
        x = nn.Dense(features=256)(x)
        x = nn.relu(x)
        lstm_state, x = nn.OptimizedLSTMCell(features=self.lstm_size)(lstm_state, x)
        x = jnp.concatenate(
            [
                nn.Dense(features=self.phi_dim)(nn.relu(nn.Dense(features=256)(x)))
                for i in range(self.num_a)
            ],
            axis=-1,
        )
        return x, lstm_state
```

The following is the CNN architecture used for ψ when ablating model capacity in VISR (22).

```
class CNNPsiMultiBranch(nn.Module):
    phi_dim: int
    num_a: int

    @nn.compact
    def __call__(self, x, w):
        x = nn.Conv(features=16, kernel_size=(3, 3))(x)
        x = nn.relu(x)
        x = x.reshape((x.shape[0], -1))  # flatten
        x = jnp.concatenate((x, w), axis=-1)
        x = nn.Dense(features=256)(x)
        x = nn.relu(x)
        x = jnp.concatenate(
            [
                nn.Dense(features=self.phi_dim)(nn.relu(nn.Dense(features=256)(x)))
                for i in range(self.num_a)
            ],
            axis=-1,
```

```
        )
        return x
```

The following is the CNN architecture used for ϕ in VISR (22).

```
class CNNPhi(nn.Module):
    output_size: int

    @nn.compact
    def __call__(self, x):
        x = nn.Conv(features=16, kernel_size=(3, 3))(x)
        x = nn.relu(x)
        x = x.reshape((x.shape[0], -1))  # flatten
        x = nn.Dense(features=256)(x)
        x = nn.relu(x)
        x = nn.Dense(features=self.output_size)(x)
        return x
```

All default parameters, model architectures, and otherwise are from the original codebase: https://github.com/mjsargent/JAX_VISR.

CSF Implementation Details

CSF Configuration		
	Description	**Default**
`lr`	Learning rate	3×10^{-4}
`batch_size`	Batch size	256
`repr_hidden_dims`	State representation network hidden dimensions	(256, 256)
`sf_hidden_dims`	Successor feature network hidden dimensions	(256, 256)
`actor_hidden_dims`	Actor network hidden dimensions	(256, 256)
`discount`	Discount factor	0.9
`tau`	Target network update rate for EMA	0.005
`skill_dim`	Skill dimension	2
`q_agg`	Aggregation for twin Q-networks ("mean" or "min")	"mean"
`target_entropy`	Target entropy action entropy ($\log \lvert\mathcal{A}\rvert$)	$\log(6)$
`num_steps`	Number of training steps	1,000,000
`episode_length`	Length of Sokoban episode, in steps	10
`grid_size`	Width and height of Sokoban grid	6

Table 2: Default training configuration details for CSF runs.

Our CSF implementation is based on an unreleased JAX implementation authored by Chongyi Zheng and Catherine Ji, which in turn is based on the OGBench (19) goal-conditioned RL evaluation benchmark. We modified this implementation using the REINFORCE(26) policy gradient to achieve gradient flow through the categorical action space.

We evaluated our CSF implementation on a modification of the Sokoban environment introduced by (5). We removed the boundary of the environment so that the agent wraps around when it moves off the edge. We modified the environment to revert to the initial state on environment reset rather than generate a new level on environment reset.

Preliminary results for CSF

We tested the effects of varying network architecture and representation dimension on the CSF representation loss. We initialized the actor, successor feature, and state representation networks with the same depth and width. We observed interesting findings when scaling encoder depth in Figure 7. With deeper networks, we see that the representation loss is driven lower for higher-dimensional skill latents. The gap between the high-dimensional skill latents widens as networks get deeper. Interestingly, we do not see the same phenomenon for wider networks.

We also studied the effect of network architecture and skill dimension on the state coverage. We calculated by sampling skills, performing `episode_length`-step rollouts with temperature 0 using each skill and counting the number of unique states visited. We tested two network shapes: one with 2 layers and one with 4 layers. Our results are presented in Figure 7. It seems that high skill

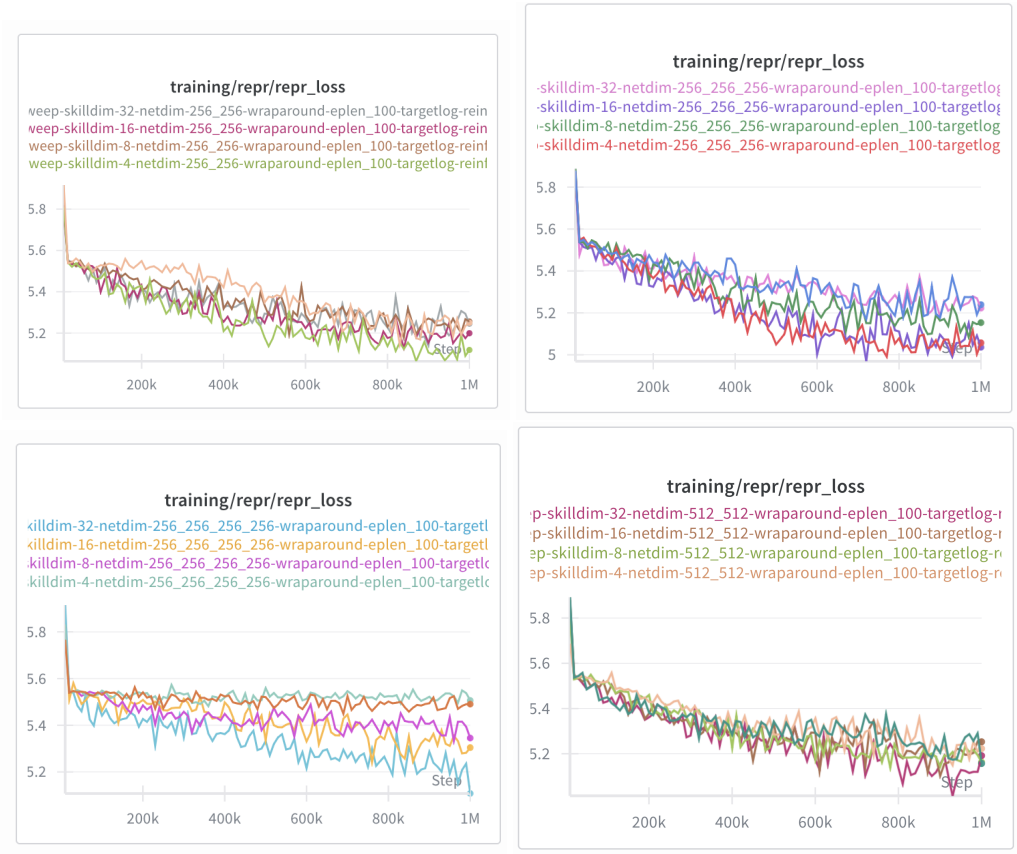

Figure 9: Representation loss for a CSF agent trained with varying network depths and skill dimension d. (Clockwise, starting at top left: width 256, depth 2; width 256, depth 3; width 512, depth 2; width 512, depth 3). Deeper networks lead to a divergence in representation loss with varying skill dimension; wider networks do not.

dimensions are important for deep networks to learn diverse trajectories. State coverage was consistently higher for high skill dimension when the network is deep, and noisier when the network is shallow. An interesting direction for future work would be training this agent with more seeds and further studying the relationship between actor network architecture and skill dimension.

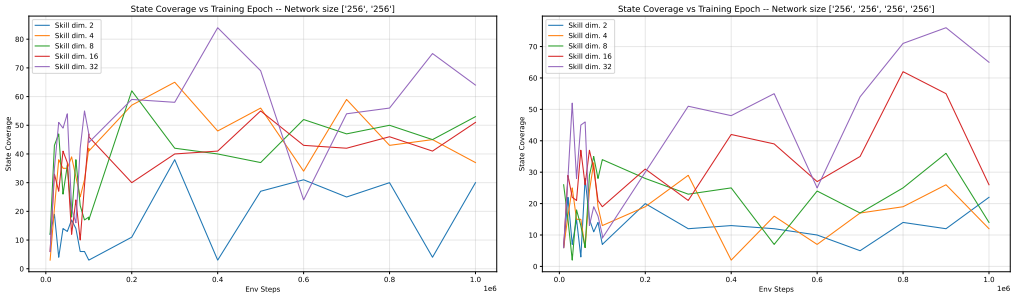

Figure 10: State coverage for a CSF agent trained with varying depths and skill dimension d. (Left to right: width 256, depth 2; width 256, depth 4). State coverage tends to increase with higher skill dimension for deep networks.