SKEW-FIT: STATE-COVERING SELF-SUPERVISED REINFORCEMENT LEARNING

Vitchyr H. Pong[*][†], **Murtaza Dalal**[*][†], **Steven Lin**[*][†], **Ashvin Nair**[†], **Shikhar Bahl**[†], **& Sergey Levine**[†]
University of California, Berkeley
{vitchyr,mdalal,stevenlin598,anair17,shikharbahl,svlevine} @berkeley.edu

ABSTRACT

In standard reinforcement learning, each new skill requires a manually-designed reward function, which takes considerable manual effort and engineering. Self-supervised goal setting has the potential to automate this process, enabling an agent to propose its own goals and acquire skills that achieve these goals. However, such methods typically rely on manually-designed goal distributions or heuristics to force the agent to explore a wide range of states. We propose a formal exploration objective for goal-reaching policies that maximizes state coverage. We show that this objective is equivalent to maximizing the entropy of the goal distribution together with goal reaching performance, where goals correspond to entire states. We present an algorithm called Skew-Fit for learning such a maximum-entropy goal distribution, and show that under certain regularity conditions, our method converges to a uniform distribution over the set of possible states, even when we do not know this set beforehand. Our experiments show that, when combined with existing goal-conditioned algorithms, Skew-Fit can learn a variety of manipulation tasks from images, including opening a door with a real robot, entirely from scratch and without any manually-designed reward function.

1 INTRODUCTION

How can we design an unsupervised reinforcement learning algorithm that automatically explores the environment and iteratively distills this experience into general-purpose policies that can accomplish new user-specified tasks at test time? For an agent to learn autonomously, it needs an exploration objective that visits as many states as possible. One way to formalize this notion in an objective is to quantify the entropy of the learned policy's state distribution $\mathcal{H}(\mathbf{S})$. However, a short-coming of this objective is that the resulting policy cannot be used to maximize user-defined rewards: such a policy only knows how to maximize state entropy. Thus, if we want to develop principled unsupervised reinforcement learning algorithms that result in useful policies, maximizing $\mathcal{H}(\mathbf{S})$ is not enough. We need a mechanism that allows us to *control* the resulting policy to achieve new goals at test-time.

In this paper, we argue that this can be accomplished by performing *goal-directed exploration*. In addition to maximizing the state distribution entropy $\mathcal{H}(\mathbf{S})$, we should be able to control where the policy goes by giving it a goal $\mathbf{G}$ that corresponds to a desired state. Mathematically, this can be accomplished by stating that a goal-conditioned policy should also minimize the conditional entropy over the states given a goal, $\mathcal{H}(\mathbf{S} \mid \mathbf{G})$. This objective provides us with a principled way for training a policy to explore all states ("maximize $\mathcal{H}(\mathbf{S})$") such that the state that is reached can be controlled by commanding goals ("minimize $\mathcal{H}(\mathbf{S} \mid \mathbf{G})$").

Directly using this objective is intractable, since it requires optimizing the marginal state distribution of the policy. However, we can avoid this difficult optimization by noting that our objective is the mutual information between the state and the goal, $I(\mathbf{S}, \mathbf{G})$, which can be written as:

$$I(\mathbf{S}, \mathbf{G}) = \mathcal{H}(\mathbf{S}) - \mathcal{H}(\mathbf{S}|\mathbf{G}) = \mathcal{H}(\mathbf{G}) - \mathcal{H}(\mathbf{G}|\mathbf{S}). \tag{1}$$

[*] Equal contribution.

[†] Berkeley AI Research, University of California, Berkeley, Computer Science

Equation 1 thus gives an equivalent objective for an unsupervised reinforcement learning algorithm: the agent should set diverse goals for itself ("maximize $\mathcal{H}(\mathbf{G})$") and learn how to reach these goals ("minimize $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$").

While the second objective—learning to reach goals—is the typical objective studied in goal-conditioned reinforcement learning (Kaelbling, 1993; Andrychowicz et al., 2017), most such methods omit the first term (Nair et al., 2018; Warde-Farley et al., 2018). However, maximizing the diversity of goals is crucial for effectively learning to reach all possible states. In an unknown environment, acquiring such a maximum-entropy goal-sampling distribution is a challenging task: how can an agent set goal states when it does not even know which states are feasible? When the states are high-dimensional, as is the case for visual observations, sampling diverse goals from the unknown manifold of valid states presents a major challenge.

In this paper, we present Skew-Fit, a method for learning to model the uniform distribution over states, given only access to data collected by an autonomous goal-conditioned policy. Skew-Fit trains a generative model on previously visited states, skewing the training data so that rarely visited states are weighted more heavily, and using density estimates from the same generative model to measure the rarity of the states.

We empirically demonstrate that, when combined with existing methods for goal-conditioned RL, Skew-Fit allows us to autonomously train goal-conditioned policies that reach diverse states. We test this method on a variety of simulated vision-based robot tasks, as well as a real-world manipulation task that requires a robot to learn to open a door without any task-specific reward function. In these experiments, Skew-Fit reaches substantially better final performance than prior methods, and learns much more quickly. We demonstrate that our approach solves the real-world door opening task from scratch in about five hours, without any manually-designed reward function.

2 Related Work

Prior work on training goal-conditioned policies assume that a goal distribution is available to sample from during exploration (Kaelbling, 1993; Schaul et al., 2015; Andrychowicz et al., 2017; Pong et al., 2018) or use a heuristics to sample goals (Colas et al., 2018b; Warde-Farley et al., 2018; Florensa et al., 2018a; Péré et al., 2018; Nair et al., 2018). Our work is complementary to these methods: rather than focusing on training goal-reaching policies, we propose a principled method for maximizing the entropy of a goal sampling distribution, $\mathcal{H}(\mathbf{G})$.

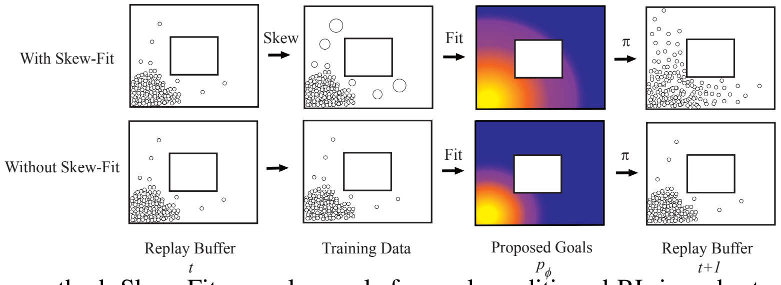


Figure 1: Our method, Skew-Fit, samples goals for goal-conditioned RL in order to induce a uniform state visitation distribution. We start by sampling from our replay buffer, and weighting the states such that rare states are given more weight. We then train a generative model $p_{\phi_{t+1}}$ with the weighted samples. By sampling new states with goals proposed from this new generative model, we obtain a wider distribution of states in our replay buffer at the next iteration. Under certain assumptions, we prove that each iteration of Skew-Fit increases the entropy of the goal distribution.

Our method stands in contrast to exploration methods that give bonus rewards (Bellemare et al., 2016; Ostrovski et al., 2017; Tang et al., 2017; Savinov et al., 2018; Chentanez et al., 2005; Lopes et al., 2012; Stadie et al., 2016; Pathak et al., 2017; Burda et al., 2018b;a; Mohamed & Rezende, 2015; Chentanez et al., 2005). These methods provide no mechanism for distilling the knowledge gained from visiting diverse states into flexible policies that can be applied to accomplish new goals at test-time: their policies visit novel states, and they quickly forget about novel states as others become more novel.

Other prior methods extract reusable skills in the form of latent-variable-conditioned policies (Hausman et al., 2018; Gupta et al., 2018b; Eysenbach et al., 2019; Gupta et al., 2018a; Florensa et al., 2017; Gregor et al., 2016). The resulting skills may be diverse, but they have no grounded interpretation, while our method can be used immediately after unsupervised training to reach diverse user-specified goals.

While some prior methods propose to choose goals based on heuristics (Baranes & Oudeyer, 2012; Veeriah et al., 2018; Colas et al., 2018a; Nachum et al., 2018; Florensa et al., 2018b), our approach provides a principled framework for optimizing a concrete and well-motivated exploration objective, and can be shown to maximize this objective under regularity assumptions.

3 Problem Formulation

Standard RL considers a Markov decision process (MDP), which has a state space $\mathcal{S}$, action space $\mathcal{A}$, and unknown dynamics $\rho(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$.Goal-conditioned RL also includes a goal space $\mathcal{G}$, which we assume to be the same as the state space, $\mathcal{G} = \mathcal{S}$. [1] [2] A goal-conditioned policy $\pi(\mathbf{a} \mid \mathbf{s}, \mathbf{g})$ maps a state $\mathbf{s} \in \mathcal{S}$ and goal $\mathbf{g} \in \mathcal{S}$ to a distribution over actions $\mathbf{a} \in \mathcal{A}$, and its objective is to reach the goal, i.e. to make the current state equal to the goal.

While most goal-conditioned RL methods (Kaelbling, 1993; Lillicrap et al., 2016; Schaul et al., 2015; Andrychowicz et al., 2017; Nair et al., 2018; Pong et al., 2018; Florensa et al., 2018a) focus on minimizing $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$ by training an accurate goal-reaching policy, we focus on the problem of setting diverse goals or, mathematically, maximizing the entropy of the goal distribution $\mathcal{H}(\mathbf{G})$. Let $U_{\mathcal{S}}$ be the uniform distribution over $\mathcal{S}$.[3] Let p_ϕ be the goal distribution, i.e. $\mathbf{G} \sim p_\phi$. Our goal is to maximize the entropy of p_ϕ, which we write as $\mathcal{H}(\mathbf{G})$. Maximizing $\mathcal{H}(\mathbf{G})$ may seem as simple as choosing the uniform distribution to be our goal distribution: $p_\phi = U_{\mathcal{S}}$. However, this requires knowing uniform distribution over valid states, which is not always trivial. In particular, we study the case where $\mathcal{S}$ is a strict, unknown subset of $\mathbb{R}^n$, for some n. For example, if the states correspond to images viewed through a robot's camera, $\mathcal{S}$ corresponds to the (unknown) set of valid images of the robot's environment, while $\mathbb{R}^n$ corresponds to all possible images, i.e. all arrays of a particular size. In such environments, sampling from the uniform distribution $\mathbb{R}^n$ is unlikely to correspond to a valid image of the real world.

We assume that we cannot sample arbitrary states from $\mathcal{S}$, but that we can sample states by performing goal-directed exploration and observing new states. To be more concrete, we introduce a simple, abstract, and somewhat simplified model of this process. First, a goal $\mathbf{G} \sim p_\phi$ is sampled from our goal distribution p_ϕ. Then, the agent attempts to achieve this goal, which results in a distribution of states $\mathbf{S} \in \mathcal{S}$ seen along the trajectory. We abstract the entire MDP episode as some generative process and write the resulting marginal distribution over $\mathbf{S}$ as $p(\mathbf{S} \mid p_\phi)$.

For the derivation and analysis of our method, we assume we have access to an oracle goal reacher, meaning that $p(\mathbf{S} \mid p_\phi) \approx p_\phi(\mathbf{S})$. In practice, we of course use goal-conditioned RL rather than an oracle. In Section 6, we demonstrate that we can combine our method with an existing goal-conditioned RL algorithm to jointly learn a goal-reaching policy and a goal sampling mechanism.

4 Skew-Fit: Learning a Maximum Entropy Goal Distribution

To learn a maximum-entropy goal proposal distribution, we present a method called Skew-Fit that iteratively increases the entropy of a generative model p_ϕ. Given a generative model p_{ϕ_t} at iteration t, we would like to train a new generative model $p_{\phi_{t+1}}$ such that $p_{\phi_{t+1}}$ has higher entropy over the set of valid states. While we do not know the set of valid states $\mathcal{S}$, we can sample states from $p(\mathbf{S} \mid p_{\phi_t})$, resulting in an empirical distribution p_{emp_t} over the states

[1] Goal-conditioned RL can always be formulated as a standard RL problem by appending the goal to the state.

[2] Some authors define the goal as a feature of the state. It is straightforward to apply the analysis and method presented in this paper to this setting.

[3] We assume $\mathcal{S}$ has finite volume so that the uniform distribution is well-defined.

$$p_{\text{emp}_t}(\mathbf{s}) = \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}\{\mathbf{s} = \mathbf{S}_n\}, \quad \mathbf{S}_n \sim p(\mathbf{S} \mid p_{\phi_t}), \tag{2}$$

and use this empirical distribution to train the next generative model $p_{\phi_{t+1}}$. However, if we simply train $p_{\phi_{t+1}}$ to model this empirical distribution, it may not necessarily have higher entropy than p_{ϕ_t}.

The intuition behind our method is quite simple: rather than fitting a generative model to our empirical distribution, we *skew* the empirical distribution so that rarely visited states are given more weight. See Figure 1 for a visualization of this process.

How should we skew the empirical distribution if we want to maximize the entropy of $p_{\phi_{t+1}}$? If we had access to the density of each state, $p_{\text{emp}_t}(\mathbf{S})$, then we could simply weight each state by $1/p_{\text{emp}_t}(\mathbf{S})$. We could then perform maximum likelihood estimation (MLE) for the uniform distribution by using the following loss to train ϕ_{t+1}:

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathbf{S}\sim U_{\mathcal{S}}}\left[\log p_\phi(\mathbf{S})\right] = \mathbb{E}_{\mathbf{S}\sim p_{\text{emp}_t}}\left[\frac{U_{\mathcal{S}}(\mathbf{S})}{p_{\text{emp}_t}(\mathbf{S})}\log p_\phi(\mathbf{S})\right] \propto \mathbb{E}_{\mathbf{S}\sim p_{\text{emp}_t}}\left[\frac{1}{p_{\text{emp}_t}(\mathbf{S})}\log p_\phi(\mathbf{S})\right]$$

where we use the fact that the uniform distribution $U_{\mathcal{S}}(\mathbf{S})$ has constant density for all states in $\mathcal{S}$. We avoid needing to model the entire MDP process, which requires an accurate model of both the dynamics and the goal-conditioned policy, by approximating $p_{\text{emp}_t}(\mathbf{S})$ with our previous learned generative model: $p_{\text{emp}_t}(\mathbf{S}) \approx p(\mathbf{S} \mid p_{\phi_t}) \approx p_{\phi_t}(\mathbf{S})$.

This procedure relies on importance sampling (IS), which can have high variance, particularly if $p_{\phi_t}(\mathbf{S}) \approx 0$. We reduce this variance by weighing each state by $p_{\phi_t}(\mathbf{S})^\alpha$, for $\alpha \in [-1, 0)$ rather than $p_\phi(\mathbf{S})^{-1}$. By choosing intermediate values of α, we can trade off the variance introduced by small $p_{\phi_t}(\mathbf{S})$ with the speed of the entropy increase to the goal distribution. Next, rather than relying on IS, we explicitly define a skewed distribution using the IS weights:

$$p_{\text{skewed}_t}(\mathbf{s}) = \frac{1}{Z_\alpha} p_{\text{emp}_t}(\mathbf{s}) p_{\phi_t}(\mathbf{s})^\alpha, \qquad \alpha \in [-1, 0), s \in \{\mathbf{S}_n\}_{n=1}^N \tag{3}$$

where Z_α is the normalizing coefficient and p_{emp_t} is given by Equation 2. Lastly, we fit the generative model at the next iteration $p_{\phi_{t+1}}$ to p_{skewed_t} using standard MLE. Shown in Section A in the Appendix is that, for a range of values of $\alpha \in [-1, 0)$, this procedure will always increase the entropy of the resulting distribution and eventually converge to a uniform distribution over valid states. We also note that because $p_{\phi_{t+1}} \approx p_{\text{skewed}_t}$, at iteration $t+1$, one can sample goals from either $p_{\phi_{t+1}}$ or p_{skewed_t}. The final Skew-Fit procedure is visualized in Figure 1 and summarized in Algorithm 1.

Algorithm 1 Skew-Fit

```
for Iteration $t = 1, 2, ...$ do
    Collect $N$ states $\{\mathbf{S}_i\}_{i=1}^N$ by sampling $G$ from $p_{\phi_t}$ (or $p_{\text{skewed}_t}$) and rolling out policy.
    Construct skewed distribution $p_{\text{skewed}_t}$ (Equation 3).
    Fit $p_{\phi_{t+1}}$ to skewed distribution $p_{\text{skewed}_t}$ using MLE.
end for
```

5 Skew-Fit with Goal-Conditioned Reinforcement Learning

Thus far, we have presented Skew-Fit assuming that we have access to an oracle goal-reaching policy. In practice we do not have access to such an oracle goal-reaching policy, and so we must combine Skew-Fit with existing goal-conditioned reinforcement learning to maximize $-\mathcal{H}(\mathbf{G} \mid \mathbf{S})$.

Maximizing $-\mathcal{H}(\mathbf{G} \mid \mathbf{S})$ requires computing the density $\log p(\mathbf{G} \mid \mathbf{S})$, which may be difficult to compute without strong modeling assumptions. However, we show in Section B that using the following reward results in maximizing a lower bound for $-\mathcal{H}(\mathbf{G} \mid \mathbf{S})$:

$$r(\mathbf{S}, \mathbf{G}) = \log q(\mathbf{G} \mid \mathbf{S}).$$

Since Skew-Fit uses a generative model to propose goals, it is particularly natural to combine with reinforcement learning with imagined goals (RIG) Nair et al. (2018), though in theory any goal-conditioned method could be used. RIG is an efficient off-policy goal-conditioned method that fits a

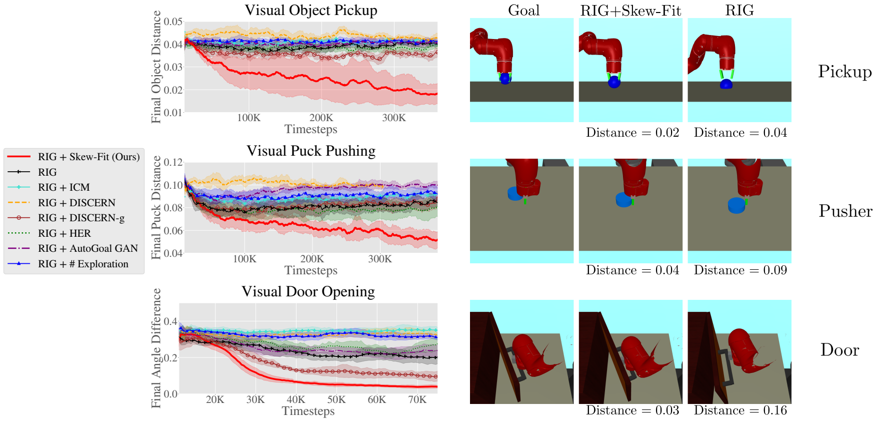


Figure 2: (Left) Learning curves for simulated continuous control experiments. Lower is better. We show the mean and standard deviation of 6 seeds and smooth temporally across 25 epochs within each seed. RIG + Skew-Fit consistently outperforms RIG and various baselines. See Appendix for description of each method. (Right) The first column displays example test goal images. The next two columns show an example image of the goal image reached by RIG + Skew-Fit and RIG. Under each image is the final distance in state space, though all tasks are trained from only images. The prior methods generally fail to generalize to these test goal images.

VAE and uses it to encode all observations and goals into a latent space. RIG also uses the generative model for both goal sampling and compute rewards, $\log q(\mathbf{G} \mid \mathbf{S})$. Applying Skew-Fit to RIG then amounts to using Skew-Fit rather than MLE to train the VAE. We also replace the underlying RL algorithm with soft actor critic (Haarnoja et al., 2018).

6 Experiments

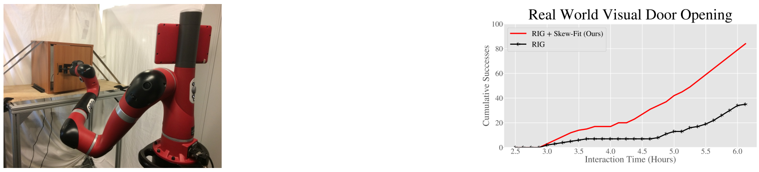


Figure 3: (Left) Picture of real-world door task setup. (Right) Learning curve for Real World Visual Door environment. We visually label a success if the policy opens the door to the target angle by the last state of the trajectory. Skew-Fit results in considerable sample efficiency gains over RIG.

Vision-based robot manipulation We evaluate Skew-Fit on simulated vision-based continuous control tasks. The agent must control a robot arm using only image observations, without access to any ground truth reward signal. Details of each environment and the chosen baselines are given in the Appendix. Training policies for these tasks is done in a completely unsupervised manner without access to any prior information about the state-space. However, to evaluate their performance, we evaluate their performance by sampling goal images from a uniform distribution. We report the final distance to the corresponding simulator state (e.g. distance of the puck to the target puck location). We see in Figure 2 that Skew-Fit significantly outperforms prior methods both in terms of task performance and sample complexity.

Real-world vision-based robotic manipulation Next, we demonstrate that Skew-Fit scales well to the real world with a door opening task.See Figure 7 for an image of the environment. We train an agent to control a Sawyer robot to open a door. We do not provide any goals to the agent and simply let it interact with the door to solve the door opening task from scratch, without any human guidance or reward signal. We train agents using **RIG + Skew-Fit** as well as **RIG**. As Figure 3 shows, standard RIG only starts to open the door consistently after five hours of training. In contrast, RIG + Skew-Fit learns to open the door after three hours of training and achieves a perfect success rate after five and a half hours of interaction time, demonstrating that Skew-Fit is a promising technique for solving real world tasks without any human-provided reward function. Videos of our method solving this task, along with the simulated environments, can be viewed on our website. [4]

Additional Experiments We also perform additional experiments including ablations and more thorough analysis in simple 2D environments in Section E.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob Mcgrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. URL https://arxiv.org/pdf/1707.01495.pdfhttp://arxiv.org/abs/1707.01495.

Adrien Baranes and Pierre-Yves Oudeyer. Active Learning of Inverse Models with Intrinsically Motivated Goal Exploration in Robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2012. doi: 10.1016/j.robot.2012.05.008. URL http://dx.doi.org/10.1016/j.robot.2012.05.008.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1471–1479, 2016.

Patrick Billingsley. *Convergence of probability measures*. John Wiley & Sons, 2013.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018a.

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

Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 1281–1288, 2005.

Cédric Colas, Pierre Fournier, Olivier Sigaud, and Pierre-Yves Oudeyer. CURIOUS: intrinsically motivated multi-task, multi-goal reinforcement learning. *CoRR*, abs/1810.06284, 2018a.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *International Conference on Machine Learning*, 2018b.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. Self-supervised Learning of Image Embedding for Continuous Control. In *Workshop on Inference to Control at NeurIPS*, 2018a.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic Goal Generation for Reinforcement Learning Agents. In *International Conference on Machine Learning (ICML)*, 2018b.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning (ICML)*, 2018.

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

[4] https://sites.google.com/view/skew-fit-scssrl

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *CoRR*, abs:1806.04640, 2018a.

Abhishek Gupta, Russell Mendonca, Yuxuan Liu, Pieter Abbeel, and Sergey Levine. Meta-Reinforcement Learning of Structured Exploration Strategies. In *Advances in Neural Information Processing Systems (NIPS)*, 2018b. URL `https://arxiv.org/pdf/1802.07245.pdf`.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

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 (ICLR)*, pp. 1–16, 2018.

L P Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume vol.2, pp. 1094 – 8, 1993.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014. URL `https://arxiv.org/pdf/1312.6114.pdf`.

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. In *International Conference on Learning Representations (ICLR)*, 2016. ISBN 0-7803-3213-X. doi: 10.1613/jair.301. URL `https://arxiv.org/pdf/1509.02971.pdf`.

Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems*, pp. 206–214, 2012.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 2125–2133, 2015.

Ofir Nachum, Google Brain, Shane Gu, Honglak Lee, and Sergey Levine. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL `https://sites.google.com/view/efficient-hrl`.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual Reinforcement Learning with Imagined Goals. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL `https://sites.google.com/site/`.

Frank Nielsen and Richard Nock. Entropies and cross-entropies of exponential families. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 3621–3624. IEEE, 2010.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International Conference on Machine Learning*, pp. 2721–2730, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. In *International Conference on Machine Learning (ICML)*, pp. 488–489. IEEE, 2017.

Alexandre Péré, Sebastien Forestier, Olivier Sigaud, and Pierre-Yves Oudeyer. Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration. In *International Conference on Learning Representations (ICLR)*, 2018. URL `https://arxiv.org/pdf/1803.00781.pdf`.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal Difference Models: Model-Free Deep RL For Model-Based Control. In *International Conference on Learning Representations (ICLR)*, 2018. URL `https://arxiv.org/pdf/1802.09081.pdf`.

Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal Value Function Approximators. In *International Conference on Machine Learning (ICML)*, pp. 1312–1320, 2015. ISBN 9781510810587. URL `http://proceedings.mlr.press/v37/schaul15.pdfhttp://jmlr.org/proceedings/papers/v37/schaul15.html`.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models. In *International Conference on Learning Representations (ICLR)*, 2016. URL `https://arxiv.org/pdf/1507.00814.pdf`.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In *Neural Information Processing Systems (NIPS)*, 2017. URL `https://arxiv.org/pdf/1611.04717.pdf`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012. ISBN 9781467317375. doi: 10.1109/IROS.2012.6386109. URL `https://homes.cs.washington.edu/{~}todorov/papers/TodorovIROS12.pdf`.

Vivek Veeriah, Junhyuk Oh, and Satinder Singh. Many-goals reinforcement learning. *arXiv preprint arXiv:1806.09605*, 2018.

David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *CoRR*, abs/1811.11359, 2018.

A Skew-Fit Analysis

In this section, we provide conditions under which p_{skewed_t} converges to the uniform distribution over the state space $\mathcal{S}$. Our most general result is stated as follows:

Lemma A.1. *Let $\mathcal{S}$ be a compact set. Define the set of distributions $\mathcal{Q} = \{p : \text{ support of } p \text{ is } \mathcal{S}\}$. Let $\mathcal{F} : \mathcal{Q} \mapsto \mathcal{Q}$ be a continuous function and such that $\mathcal{H}(\mathcal{F}(p)) \geq \mathcal{H}(p)$ with equality if and only if p is the uniform probability distribution on $\mathcal{S}$, $U_{\mathcal{S}}$. Define the sequence of distributions $P = (p_1, p_2, \dots)$ by starting with any $p_1 \in \mathcal{Q}$ and recursively defining $p_{t+1} = \mathcal{F}(p_t)$.*

The sequence P converges to $U_{\mathcal{S}}$.

Proof. See C.1. □

The assumption that $\mathcal{S}$ is compact is easily achieved in most application and makes $U_{\mathcal{S}}$ well defined.

We will apply Lemma A.1 to be the map from p_{skewed_t} to $p_{\text{skewed}_{t+1}}$ to show that p_{skewed_t} converges to $U_{\mathcal{S}}$[5]. Skew-Fit produces a sequence of distributions $(p_{\phi_1}, p_{\text{emp}_1}, p_{\text{skewed}_1}, p_{\phi_2}, \dots)$, and so we need to reason about the intermediate distributions, p_{ϕ_t} and p_{emp_t}. We begin with a few assumptions about the optimization method that maps p_{skewed_t} to $p_{\phi_{t+1}}$ and the goal-conditioned policy that maps p_{ϕ_t} to p_{emp_t}, which are subroutines in Skew-Fit. We assume that these maps are continuous and do not decrease the entropy, i.e. $\mathcal{H}(p_{\text{emp}_{t+1}}) \geq \mathcal{H}(p_{\phi_{t+1}}) \geq \mathcal{H}(p_{\text{skewed}_t})$. The continuity assumption states that the method used to fit $p_{\phi_{t+1}}$ and the goal-conditioned policy are well-behaved. Moreover, making a statement without the entropy assumption would be difficult: if the goal-conditioned policy ignored the goal and always entered the same state, or if the generative model only captured a single mode p_{skewed_t}, it would be challenging for any procedure to result in the uniform distribution.

Next, we assume the support of p_{ϕ_t} contains $\mathcal{S}$, so that p_{skewed_t} is well defined and a continuous function of p_{emp_t} and p_{ϕ_t}. Note that p_{ϕ_t} can have support larger than $\mathcal{S}$, and so we can choose to optimize p_{ϕ_t} over any class of generative models with wide supports, without needing to know the manifold $\mathcal{S}$.

To use Lemma A.1, we must show $\mathcal{H}(p_{\text{skewed}_t}) \geq \mathcal{H}(p_{\text{emp}_t})$ with equality if and only if $p_{\text{emp}_t} = U_{\mathcal{S}}$. For the simple case when $p_{\phi_t} = p_{\text{emp}_t}$ identically at each iteration, we prove that this is true for any value of $\alpha \in [-1, 0)$ in Lemma C.2 of the Appendix.

The entropy of p_{skewed_t} becomes more difficult to analyze when $p_{\phi_t} \neq p_{\text{emp}_t}$. However, we prove the following result:

Lemma A.2. *Given two distribution p_{emp_t} and p_{ϕ_t} where $p_{emp_t} \ll p_{\phi_t}$ [6] and*

$$0 < \mathrm{Cov}_{\mathbf{S} \sim p_{emp_t}} \left[\log p_{emp_t}(\mathbf{S}), \log p_{\phi_t}(\mathbf{S}) \right], \tag{4}$$

define the distribution p_{skewed_t} as in Equation 3. Let $\mathcal{H}_\alpha(\alpha)$ be the entropy of p_{skewed_t} for a fixed α. Then there exists a constant $a < 0$ such that for all $\alpha \in [a, 0)$,

$$\mathcal{H}(p_{skewed_t}) = \mathcal{H}_\alpha(\alpha) > \mathcal{H}(p_{emp_t}).$$

Proof. See C.4 □

While Lemma A.2 does not give an exact value for α, it states that if we choose negative values of α that are small enough and if the log densities of p_{emp_t} and p_{ϕ_t} are positively correlated, then we can guarantee that the entropy of p_{skewed_t} will be higher then the entropy of p_{emp_t}. In practice, we expect the correlation to be frequently positive with an accurate goal-conditioned policy, since p_{emp_t} is the set of states seen when trying to reach goals from p_{ϕ_t}. Moreover, we found that α values as low as $\alpha = -1$ performed well. Lastly, the condition in Equation 4 is impossible to achieve if and only if $\log p_{\text{emp}_t}(\mathbf{S})$ is a constant, meaning that p_{emp_t} is the uniform distribution.

In summary, we see that under certain assumptions, p_{skewed_t} converges to $U_{\mathcal{S}}$. Since we train each generative model $p_{\phi_{t+1}}$ by fitting it to p_{skewed_t}, we expect p_{ϕ_t} to also converge to $U_{\mathcal{S}}$. We verify this numerically on both toy domains and realistic RL problems in our experiments.

[5] We take $N \to \infty$ and refer to convergence in distribution.

[6] $p \ll q$ means that p is absolutely continuous with respect to q, i.e. $p(\mathbf{s}) = 0 \implies q(\mathbf{s}) = 0$.

B Policy Reward Derivation

Maximizing $I(\mathbf{G}, \mathbf{S})$ can be done by simultaneously performing Skew-Fit and training a goal conditioned policy to minimize $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$, or, equivalently, maximize $-\mathcal{H}(\mathbf{G} \mid \mathbf{S})$. Maximizing $-\mathcal{H}(\mathbf{G} \mid \mathbf{S})$ requires computing the density $\log p(\mathbf{G} \mid \mathbf{S})$, which may be difficult to compute without strong modeling assumptions. However, for any distribution q, the following lower bound for $-\mathcal{H}(\mathbf{G} \mid \mathbf{S})$ holds:

$$\begin{aligned}-\mathcal{H}(\mathbf{G} \mid \mathbf{S}) &= \mathbb{E}_{(\mathbf{G},\mathbf{S})\sim p_{\phi_t},\pi}\left[\log q(\mathbf{G} \mid \mathbf{S})\right] + D_{\mathrm{KL}}(p \mid q)\\ &\geq \mathbb{E}_{(\mathbf{G},\mathbf{S})\sim p_{\phi_t},\pi}\left[\log q(\mathbf{G} \mid \mathbf{S})\right]\end{aligned}$$

where D_{KL} denotes Kullback–Leibler divergence. Thus to minimize $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$, we train a policy to maximize the following reward:

$$r(\mathbf{S}, \mathbf{G}) = \log q(\mathbf{G} \mid \mathbf{S}).$$

C Proofs

Let $q \ll p$ means that q is absolutely continuous with respect to p, i.e. $p(x) = 0 \implies q(x) = 0$.

Lemma C.1. *Let $\mathcal{S}$ be a compact set. Define the set of distributions $\mathcal{Q} = \{p : \text{ support of } p \text{ is } \mathcal{S}\}$. Let $\mathcal{F} : \mathcal{Q} \mapsto \mathcal{Q}$ be a continuous function and such that $\mathcal{H}(\mathcal{F}(p)) \geq \mathcal{H}(p)$ with equality if and only if p is the uniform probability distribution on $\mathcal{S}$, $U_{\mathcal{S}}$. Define the sequence of distributions $P = (p_1, p_2, \dots)$ by starting with any $p_1 \in \mathcal{Q}$ and recursively defining $p_{t+1} = \mathcal{F}(p_t)$.*

The sequence P converges to $U_{\mathcal{S}}$.

Proof. The uniform distribution $U_{\mathcal{S}}$ is well defined since $\mathcal{S}$ is compact. Because $\mathcal{S}$ is a compact set, by Prokhorov's Theorem Billingsley (2013), the set $\mathcal{Q}$ is sequentially compact. Thus, P has a convergent subsequence $P' = (p_{k_1}, p_{k_2}, \dots) \subset P$ for $k_1 < k_2 < \dots$ that converges to a distribution $p^* \in \mathcal{Q}$. Because $\mathcal{F}$ is continuous, p^* must be a fixed point of $\mathcal{F}$ since by the convergence mapping theorem, we have that

$$\lim_{i\to\infty} p_{k_i} = p^* \implies \lim_{i\to\infty} \mathcal{F}(p_{k_i}) = \mathcal{H}(p^*)$$

and so

$$\begin{aligned}p^* &= \lim_{i\to\infty} p_{k_i}\\ &= \lim_{i\to\infty} \mathcal{F}(p_{k_{i-1}})\\ &= \mathcal{H}(p^*).\end{aligned}$$

The only fixed point of $\mathcal{F}$ is $U_{\mathcal{S}}$ since for any distribution p that is not the uniform distribution, $U_{\mathcal{S}}$, we have that $\mathcal{H}(\mathcal{F}(p)) > \mathcal{H}(p)$ which implies that $\mathcal{F}(p) \neq p$. Thus, P' converges to the only fixed point, $U_{\mathcal{S}}$. Since the entropy cannot decrease, then entropy of the distributions in P must also converge to the entropy of $U_{\mathcal{S}}$. Lastly, since entropy is a continuous function of distribution, P must converge to $U_{\mathcal{S}}$. □

Lemma C.2. *Assume the set $\mathcal{S}$ has finite volume so that its uniform distribution $U_{\mathcal{S}}$ is well defined and has finite entropy. Given any distribution $p(\mathbf{s})$ whose support is $\mathcal{S}$, recursively define p_α*

$$p_\alpha(\mathbf{s}) = \frac{1}{Z_\alpha} p(\mathbf{s})^\alpha, \quad \forall \mathbf{s} \in \mathcal{S}$$

where Z_α is the normalizing constant and $\alpha \in [0, 1)$[7].

[7] In the paper, $\alpha \in [-1, 0)$. However, when $p_{\mathrm{emp}_t} = p_{\phi_t}$, Equation 3 becomes

$$\begin{aligned}p_{\mathrm{skewed}_t}(\mathbf{S}) &= \frac{1}{Z_\alpha} p_{\phi_t}(\mathbf{S}) p_{\phi_t}(\mathbf{S})^\alpha, \qquad \alpha \in [-1, 0)\\ &= \frac{1}{Z_\alpha} p_{\phi_t}(\mathbf{S})^{\alpha'}, \qquad \alpha' \in [0, 1)\end{aligned}$$

For all $\alpha \in [0, 1)$,

$$\mathcal{H}(p_\alpha) \geq \mathcal{H}(p)$$

with equality if and only if p is $U_\mathcal{S}$, the uniform distribution $\mathcal{S}$.

Proof. If $\alpha = 0$ or p is the uniform distribution, the result is clearly true. We now study the case where $\alpha \in (0, 1)$ and $p \neq U_\mathcal{A}$.

Define the one-dimensional exponential family $\{p_\theta^t : \alpha \in [0, 1]\}$ where p_θ^t is

$$p_\theta^t(\mathbf{s}) = e^{\alpha T(\mathbf{s}) - A(\alpha) + k(\mathbf{s})}$$

with log carrier density $k(\mathbf{s}) = 0$, natural parameter α, sufficient statistic $T(\mathbf{s}) = \log p_t(\mathbf{s})$, and log-normalizer $A(\alpha) = \int_\mathcal{S} e^{\alpha T(\mathbf{s})} d\mathbf{s}$. As shown in Nielsen & Nock (2010), the entropy of a distribution from a one-dimensional exponential family with parameter α is given by:

$$\mathcal{H}_\theta^t(\alpha) \triangleq \mathcal{H}(p_\theta^t) = A(\alpha) - \alpha A'(\alpha)$$

The derivative with respect to α is then

$$\begin{aligned}
\frac{d}{d\alpha} d\mathcal{H}_\theta^t(\alpha) &= -\alpha A''(\alpha) \\
&= -\alpha \mathrm{Var}_{\mathbf{s} \sim p_\theta^t}[T(\mathbf{s})] \\
&= -\alpha \mathrm{Var}_{\mathbf{s} \sim p_\theta^t}[\log p_t(\mathbf{s})] \\
&\leq 0
\end{aligned}$$

where we use the fact that the nth derivative of $A(\alpha)$ is the n central moment, i.e. $A''(\alpha) = \mathrm{Var}_{\mathbf{s} \sim p_\theta^t}[T(\mathbf{s})]$. Since variance is always non-negative, this means the entropy is monotonically decreasing with α, and so

$$\mathcal{H}(p_\alpha) \geq \mathcal{H}(p_1) = \mathcal{H}(p)$$

with equality if and only if

$$\mathrm{Var}_{\mathbf{s} \sim p_\theta^t}[\log p(\mathbf{s})] = 0.$$

However, this only happens if $\log p(\mathbf{s})$ is constant over its support, i.e. it is the uniform distribution over its support. □

We also prove the convergence directly for the (even more) simplified case when $p_{\text{skewed}_t} = p_{\phi_{t+1}} = p_{\text{emp}_{t+1}}$ using a similar technique:

Lemma C.3. *Assume the set $\mathcal{S}$ has finite volume so that its uniform distribution $U_\mathcal{S}$ is well defined and has finite entropy. Given any distribution $p(\mathbf{s})$ whose support is $\mathcal{S}$, recursively define p_t with $p_1 = p$ and*

$$p_{t+1}(\mathbf{s}) = \frac{1}{Z_\alpha^t} p_t(\mathbf{s})^\alpha, \quad \forall \mathbf{s} \in \mathcal{S}$$

where Z_α^t is the normalizing constant and $\alpha \in [0, 1)$.

The sequence $(p_1, p_2, \dots)$ converges to $U_\mathcal{S}$, the uniform distribution $\mathcal{S}$.

Proof. If $\alpha = 0$, then p_2 (and all subsequent distributions) will clearly be the uniform distribution. We now study the case where $\alpha \in (0, 1)$.

At each iteration t, define the one-dimensional exponential family $\{p_\theta^t : \theta \in [0, 1]\}$ where p_θ^t is

$$p_\theta^t(\mathbf{s}) = e^{\theta T(\mathbf{s}) - A(\theta) + k(\mathbf{s})}$$

with log carrier density $k(\mathbf{s}) = 0$, natural parameter θ, sufficient statistic $T(\mathbf{s}) = \log p_t(\mathbf{s})$, and log-normalizer $A(\theta) = \int_\mathcal{S} e^{\theta T(\mathbf{s})} d\mathbf{s}$. As shown in Nielsen & Nock (2010), the entropy of a distribution from a one-dimensional exponential family with parameter θ is given by:

$$\mathcal{H}_\theta^t(\theta) \triangleq \mathcal{H}(p_\theta^t) = A(\theta) - \theta A'(\theta)$$

The derivative with respect to θ is then

$$\begin{aligned}\frac{d}{d\theta} d\mathcal{H}^t_\theta(\theta) &= -\theta A''(\theta) \\ &= -\theta \mathrm{Var}_{\mathbf{s} \sim p^t_\theta}[T(\mathbf{s})] \\ &= -\theta \mathrm{Var}_{\mathbf{s} \sim p^t_\theta}[\log p_t(\mathbf{s})] \\ &\leq 0\end{aligned} \tag{5}$$

where we use the fact that the nth derivative of $A(\theta)$ is the n central moment, i.e. $A''(\theta) = \mathrm{Var}_{\mathbf{s} \sim p^t_\theta}[T(\mathbf{s})]$. Since variance is always non-negative, this means the entropy is monotonically decreasing with θ. Note that p_{t+1} is a member of this exponential family, with parameter $\theta = \alpha \in (0, 1)$. So

$$\mathcal{H}(p_{t+1}) = \mathcal{H}^t_\theta(\alpha) \geq \mathcal{H}^t_\theta(1) = \mathcal{H}(p_t)$$

which implies

$$\mathcal{H}(p_1) \leq \mathcal{H}(p_2) \leq \dots .$$

This monotonically increasing sequence is upper bounded by the entropy of the uniform distribution, and so this sequence must converge.

The sequence can only converge if $\frac{d}{d\theta}\mathcal{H}^t_\theta(\theta)$ converges to zero. However, because α is bounded away from 0, Equation 5 states that this can only happen if

$$\mathrm{Var}_{\mathbf{s} \sim p^t_\theta}[\log p_t(\mathbf{s})] \to 0. \tag{6}$$

Because p_t has full support, then so does p^t_θ. Thus, Equation 6 is only true if $\log p_t(\mathbf{s})$ converges to a constant, i.e. p_t converges to the uniform distribution. □

Lemma C.4. *Given two distribution $p(x)$ and $q(x)$ where $p \ll q$ and*

$$0 < \mathrm{Cov}_p[\log p(X), \log q(X)] \tag{7}$$

define the distribution p_α as

$$p_\alpha(x) = \frac{1}{Z_\alpha} p(x) q(x)^\alpha$$

where $\alpha \in \mathbb{R}$ and Z_α is the normalizing factor. Let $\mathcal{H}_\alpha(\alpha)$ be the entropy of p_α. Then there exists a constant $a > 0$ such that for all $\alpha \in [-a, 0)$,

$$\mathcal{H}_\alpha(\alpha) > \mathcal{H}_\alpha(0) = \mathcal{H}(p). \tag{8}$$

Proof. Observe that $\{p_\alpha : \alpha \in [-1, 0]\}$ is a one-dimensional exponential family

$$p_\alpha(x) = e^{\alpha T(x) - A(\alpha) + k(x)}$$

with log carrier density $k(x) = \log p(x)$, natural parameter α, sufficient statistic $T(x) = \log q(x)$, and log-normalizer $A(\alpha) = \int_\mathcal{X} e^{\alpha T(x) + k(x)} dx$. As shown in Nielsen & Nock (2010), the entropy of a distribution from a one-dimensional exponential family with parameter α is given by:

$$\mathcal{H}_\alpha(\alpha) \triangleq \mathcal{H}(p_\alpha) = A(\alpha) - \alpha A'(\alpha) - \mathbb{E}_{p_\alpha}[k(X)]$$

The derivative with respect to α is then

$$\begin{aligned}\frac{d}{d\alpha}\mathcal{H}_\alpha(\alpha) &= -\alpha A''(\alpha) - \frac{d}{d\alpha}\mathbb{E}_{p_\alpha}[k(x)] \\ &= -\alpha A''(\alpha) - \mathbb{E}_\alpha[k(x)(T(x) - A'(\alpha)] \\ &= -\alpha \mathrm{Var}_{p_\alpha}[T(x)] - \mathrm{Cov}_{p_\alpha}[k(x), T(x)]\end{aligned}$$

where we use the fact that the nth derivative of $A(\alpha)$ give the n central moment, i.e. $A'(\alpha) = \mathbb{E}_{p_\alpha}[T(x)]$ and $A''(\alpha) = \mathrm{Var}_{p_\alpha}[T(x)]$. The derivative of $\alpha = 0$ is

$$\begin{aligned}\frac{d}{d\alpha}\mathcal{H}_\alpha(0) &= -\mathrm{Cov}_{p_0}[k(x), T(x)] \\ &= -\mathrm{Cov}_p[\log p(x), \log q(x)]\end{aligned}$$

which is negative by assumption. Because the derivative at $\alpha = 0$ is negative, then there exists a constant $a > 0$ such that for all $\alpha \in [-a, 0]$, $\mathcal{H}_\alpha(\alpha) > \mathcal{H}_\alpha(0) = \mathcal{H}(p)$. □

D Environment Details

Point-Mass: In this environment, an agent must learn to navigate a square-shaped corridor (see Figure 6). The observation is the 2D position, and the agent must specify a velocity as the 2D action. The reward at each time step is the negative distance between the achieved position and desired position.

Visual Pusher: A MuJoCo environment with a 7-DoF Sawyer arm and a small puck on a table that the arm must push to a target position. The agent controls the arm by commanding x, y position for the end effector (EE). The underlying state is the EE position, e and puck position p. The evaluation metric is the distance between the goal and final puck positions. The hand goal/state space is a 10x10 cm^2 box and the puck goal/state space is a 30x20 cm^2 box. Both the hand and puck spaces are centered around the origin. The action space ranges in the interval $[-1, 1]$ in the x and y dimensions.

Visual Door: A MuJoCo environment with a 7-DoF Sawyer arm and a door on a table that the arm must pull open to a target angle. Control is the same as in *Visual Pusher*. The evaluation metric is the distance between the goal and final door angle, measured in radians. In this environment, we do not reset the position of the hand or door at the end of each trajectory. The state/goal space is a 5x20x15 cm^3 box in the x, y, z dimension respectively for the arm and an angle between $[0, .83]$ radians. The action space ranges in the interval $[-1, 1]$ in the x, y and z dimensions.

Visual Pickup: A MuJoCo environment with the same robot as *Visual Pusher*, but now with a different object. The object is cube-shaped, but a larger intangible sphere is overlaid on top so that it is easier for the agent to see. Moreover, the robot is constrained to move in 2 dimension: it only controls the y, z arm positions. The x position of both the arm and the object is fixed. The evaluation metric is the distance between the goal and final object position. For the purpose of evaluation, 75% of the goals have the object in the air and 25% have the object on the ground. The state/goal space for both the object and the arm is 10cm in the y dimension and 13cm in the z dimension. The action space ranges in the interval $[-1, 1]$ in the y and z dimensions.

Real World Visual Door: A Rethink Sawyer Robot with a door on a table. The arm must pull the door open to a target angle. The agent controls the arm by commanding the x, y, z velocity of the EE. Our controller commands actions at a rate of up to 10Hz with the scale of actions ranging up to 1cm in magnitude. The underlying state and goal is the same as in *Visual Door*. Again we do not reset the position of the hand or door at the end of each trajectory. We obtain images using a Kinect Sensor and our robotic control code can be found at `https://github.com/mdalal2020/sawyer_control.git` The state/goal space for the environment is a 10x10x10 cm^3 box. The action space ranges in the interval $[-1, 1]$ (in cm) in the x, y and z dimensions. The door angle lies in the range $[0, 30]$ degrees.

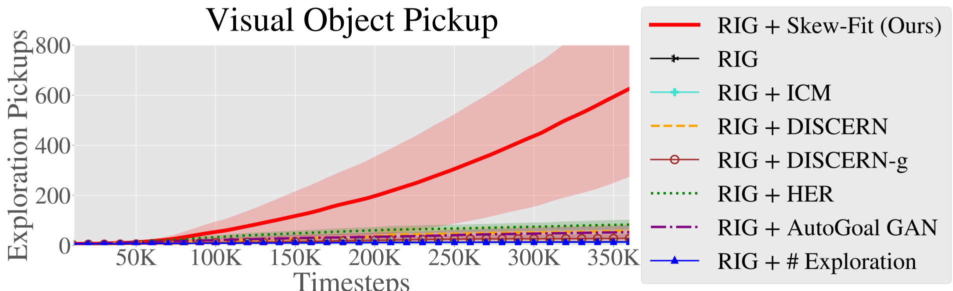


Figure 5: Cumulative total pickups during exploration for each method. RIG + Skew-Fit quickly learns to learn to pick up the object while the baselines fail to pay attention to the object.

E Additional Experiment and Experiment Details

E.1 Baselines

First, we compare to standard **RIG**, without Skew-Fit. We also compare to hindsight experience replay (HER) Andrychowicz et al. (2017), which relabels goals based on states seen in the rest of the trajectory. While the original HER paper operates directly on the raw state space, we were unable to get HER to learn from pixels. We instead ran HER on the same latent space used as our method, and use the learned generative model to sample goals for exploration. We denote this baseline **RIG + HER**. Florensa et al. (2018b) samples goals from a GAN based on the difficulty of reaching the goal. We include a comparison against this method by replacing p_ϕ with the GAN and label it **RIG + AutoGoal**. We compare to Warde-Farley et al. (2018), which uses a non-parametric approach based on clustering to sample goals and a state discriminator to compute rewards. When trained either on images or the RIG latent state, we were unable to obtain good results, and have included the latter as **RIG + DISCERN**. We also compare to the goal proposal mechanism proposed by Warde-Farley et al. (2018) without the discriminiative reward in DISCERN, which we label **RIG + DISCERN-g**. Lastly, we compare our method to two exploration methods based on reward bonuses: ICM (Pathak et al., 2017), which rewards an agent for visiting states that are difficult to predict, and # Exploration (Tang et al., 2017), which rewards an agent for visiting novel states, where novelty is measured using a hash table. These two baselines are denoted **RIG + ICM** and **RIG + #Exploration** respectively.

E.2 2D Navigation Experiments

We initialize the VAE to only output points in the bottom left corner of the environment. Both the encoder and decoder have ReLU hidden activations, 2 hidden layers with 32 units, and no output activations. The VAE has a latent dimension of 16 and a Gaussian decoder trained with mean-squared error loss, batch size of 500, and 100 epochs per iteration. For Skew-Fit hyperparameters, $\alpha = -0.5$ and $N = 10000$.

For the RL version of this task, the VAE was trained in the same way. The RL hyperparameters are listed in Table 1. Our experimental evaluation of Skew-Fit aims to study the following empirical questions: **(1)** Can Skew-Fit learn a generative model to find the uniform distribution over the set of valid states? **(2)** Does Skew-Fit work on high dimensional state spaces, such as images? **(3)** When combined with goal-conditioned RL, can Skew-Fit enable agents to autonomously set and learn to reach a diverse set of goals?

Section E.3 studies the first question in the context of a simple 2-dimensional navigation environment. Section E.4 studies the second question by applying Skew-Fit to both simulated and real-world images in an unsupervised learning setting, without a goal-conditioned policy. Finally, Section **??** analyzes the performance of Skew-Fit when combined with RIG on a variety of simulated tasks, vision-based robot manipulation tasks.

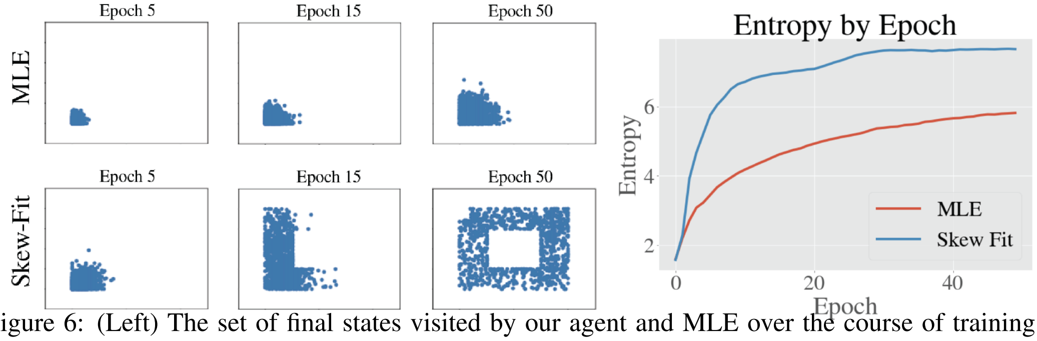


Figure 6: (Left) The set of final states visited by our agent and MLE over the course of training. In contrast to MLE, our method quickly approaches a uniform distribution over the set of valid states. (Right) The entropy of the sample data distribution, which quickly reaches its maximum for Skew-Fit. The entropy was calculated via discretization onto a 60 by 60 grid.

E.3 Skew-Fit on Simplified RL Environment

We first analyze the effect of Skew-Fit for learning a goal distribution in isolation from training a goal-reaching policy. To this end, we study an idealized example where the policy is a hand-designed, near-perfect goal-reaching policy.

The MDP is defined on a 2-by-2 unit square-shaped corridor (see Figure 6). At the beginning of an episode, the agent begins in the bottom-left corner and samples a goal from a goal distribution p_{ϕ_t}. To model an imperfect policy, we add zero-mean Gaussian noise to this sampled goal with a standard deviation of 0.05. The policy reaches the state that is closest to this noisy goal and inside the corridor, giving us a state $\mathbf{S}$ to add to our empirical distribution. After collecting $N = 10000$ states using the process above, we train $p_{\phi_{t+1}}$ on the collected states and then repeat the entire procedure, this time sampling goals from $p_{\phi_{t+1}}$. We compare Skew-Fit to a goal sampling distribution that is only trained using maximum likelihood estimation (MLE).

As seen in Figure 6, Skew-Fit results in learning a high entropy, near-uniform distribution over the state space. In contrast, MLE only models the states that are explored by the initial noise of the policy, resulting in the policy only setting goals in and exploring the bottom left corner. These results empirically validate that naively using previous experience to set goals will not result in diverse exploration and that Skew-Fit results in a maximum-entropy goal-sampling distribution.

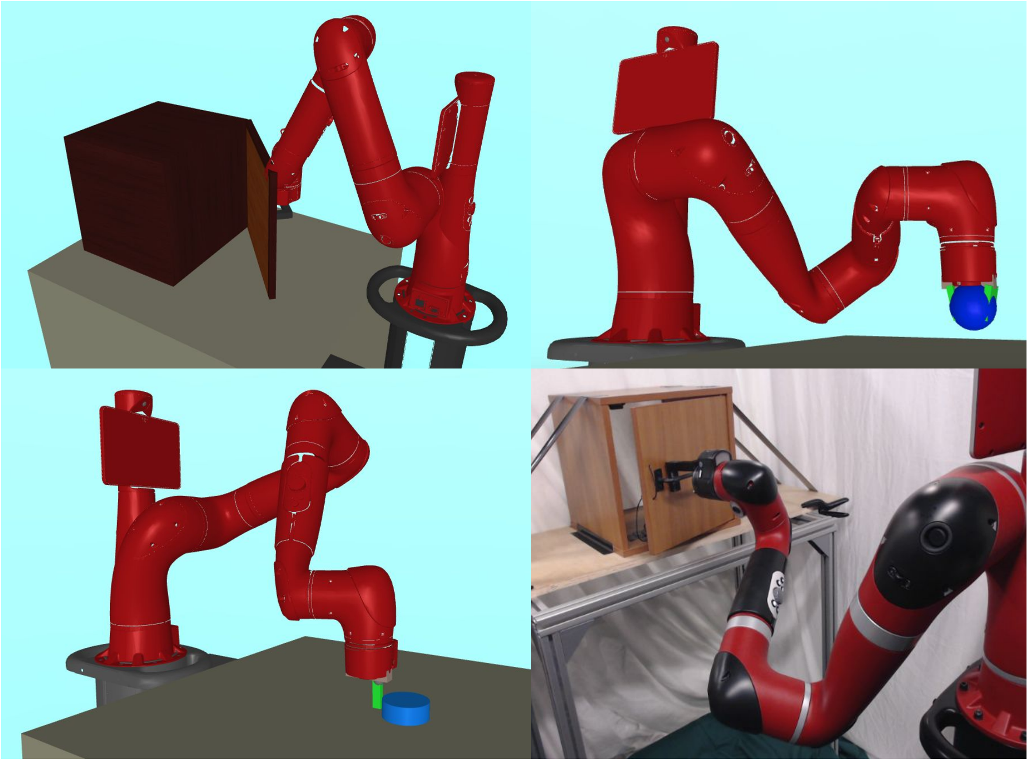
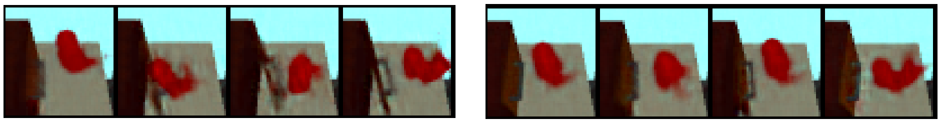


Figure 7: Here we display all four of our continuous control environments. In the top left corner, *Visual Door*, the simulated door opening environment. In the top right, *Visual Pickup*, the simulated object pick up task. The bottom left display *Visual Pusher*, the simulated puck pushing environment while the bottom right is *Real World Visual Door*, the real world door opening task. See appendix for more details.



(a) Skew-Fit

(b) MLE

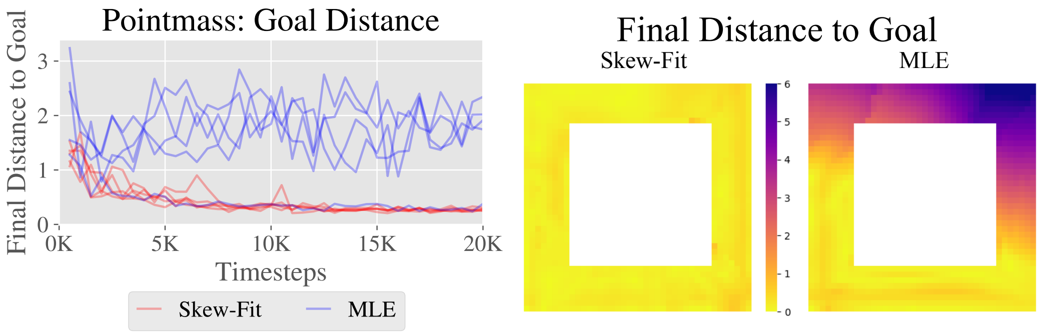
Figure 8: Samples from a generative model p_ϕ when trained with (a) Skew-Fit and with (b) maximum likelihood estimation trained on images from the simulated door opening task. The models are trained on a dataset collected by executing a random policy in the environment, which results in mostly images with a closed door and only occasional images with the door open. Note that the Skew-Fit samples are substantially more diverse, meaning that if p_ϕ were used to sample goals, it would encourage the agent to practice opening the door more frequently.

E.4 Modeling Visual Observations

We would like to use Skew-Fit to learn maximum-entropy distributions over complex, high-dimensional state spaces, where we cannot manually design these uniform distributions. The next set of experiments study how Skew-Fit can be used to train a generative model to sample diverse images when trained on an imbalanced dataset. For these experiments, we use a simulated MuJoCo (Todorov et al., 2012) environment and real environment that each consists of a 7 degree of freedom robot arm in front of a door that it can open. See Figure 7 for a visualization of the simulated and real-world door environment, and the Appendix for more details on the environment.

We generate a dataset of images from the environment by running a policy that samples actions uniformly at random. Such a policy represents a particularly challenging setting for standard VAE training methods: a policy that chooses random actions does not visit uniformly random states, but rather states that are heavily biased toward ones that resemble the initial state. In the door opening environment, this means that many of the samples have the door in the initial closed position, and only the robot's arm moves. We then train two generative models on these datasets: one using Skew-Fit and another using MLE. For our generative model, we use the same generative model as the one in RIG, a variational autoencoder Kingma & Welling (2014). To estimate the likelihood of our data, we use Monte Carlo estimation and importance sampling to marginalize the latent variables. See Appendix Section E.7 for experimental details.

In this experiment, we do not train a goal-conditioned policy, and instead only study how Skew-Fit can be used to effectively "balance" this dataset. As can be seen in Figure 8 and Figure **??**, the samples produced by the model trained with Skew-Fit generally have a much wider range of door angles, while the model trained with MLE only captures a single mode of the door, both for simulated and real-world images.



(a) (b)

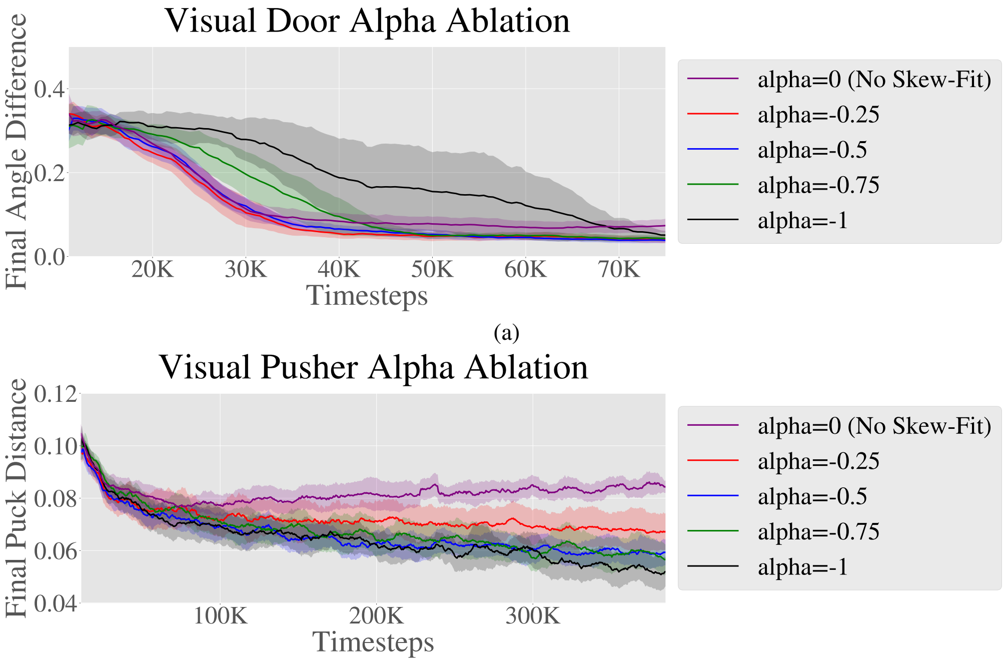
Figure 9: (a) Comparison of Skew-Fit vs MLE goal sampling on final distance to goal on RL version of the pointmass environment. Skew-Fit consistently learns to solve the task, while MLE often fails. (b) Heatmaps of final distance to each possible goal location for Skew-Fit and MLE. Skew-Fit learns a good policy over the entire state space, but MLE performs poorly for states far away from the starting position.

E.5 2D navigation

We now provide a simple experiment that combines Skew-Fit with a goal-conditioned policy that is trained. We reproduce the 2D navigation environment experiment from Section E.3, and replace the oracle goal-reacher with a goal-reaching policy that is simultaneously trained. The policy outputs velocities with maximum speed of one. Evaluation goals are chosen uniformly over the valid states. In Figure 9a, we can see that a policy trained with a goal distribution trained by Skew-Fit consistently learns to reach all goals, whereas a goal distribution trained with MLE results in a policy that fails to reach states far from the starting position.

E.6 Sensitivity Analysis

We study the sensitvity of the α hyperparameter by testing values of $\alpha \in [0, -0.25, -0.5, -0.75, -1]$ on the Visual Door and Visual Pusher task. The results are included in the Appendix in Figure 10 and shows that our method is relatively robust to different parameters of α, particularly for the more challenging Visual Pusher task.



(a)

(b)

Figure 10: We sweep different values of α on (a) Visual Door and (b) Visual Pusher. Skew-Fit helps the finally performance marginally on the Visual Door task, and even degrades performance if α is large in magnitude. In the more challenging Visual Pusher task, we see that Skew-Fit consistently helps and halves the final distance.

E.7 Vision-Based Continuous Control Experiments

For our underlying RL algorithm, we use a modified version of soft actor critic (SAC) with automated entropy tuning Haarnoja et al. (2018) and twin Q-functions Fujimoto et al. (2018). This is in contrast to the original RIG Nair et al. (2018) paper which used TD3 Fujimoto et al. (2018). We found that maximum entropy policies in general improved the performance of RIG, and that we did not need to add noise on top of the stochastic policy's noise. For our RL network architectures and training scheme, we use fully connected networks for the policy, Q-function and value networks with two hidden layers of size 400 and 300 each. We also delay training any of these networks for 10000 time steps in order to collect sufficient data for the replay buffer as well as to ensure the latent space of the VAE is relatively stable (since we train the VAE online in this setting). As in RIG, we train a goal-conditioned value functions Schaul et al. (2015) using hindsight experience replay Andrychowicz et al. (2017), relabelling 50% of exploration goals as goals sampled from the VAE prior $\mathcal{N}(0, 1)$ and 30% from future goals in the trajectory.

In our experiments, we use an image size of 48x48. For our VAE architecture, we use a modified version of the architecture used in the original RIG paper Nair et al. (2018). Our VAE has three convolutional layers with kernel sizes: 5x5, 3x3, and 3x3, number of output filters: 16, 32, and 64 and strides: 3, 2, and 2. We then have a fully connected layer with the latent dimension number of units, and then reverse the architecture with de-convolution layers. We vary the latent dimension of the VAE, the β term of the VAE and the α term for Skew-Fit based on the environment. Additionally, we vary the training schedule of the VAE based on the environment. See the table at the end of the

Hyper-parameter	**Value**
Algorithm	TD3 Fujimoto et al. (2018)[a]
# training batches per time step	1
Q network hidden sizes	$400, 300$
Policy network hidden sizes	$400, 300$
Q network and policy activation	ReLU
Exploration Noise	None
RL Batch Size	1024
Discount Factor	0.99
Path length	25
Reward Scaling	100
Number of steps per epoch	5000

Table 1: Hyper-parameters used for 2D RL experiment (Figure 9a).

[a] We expect similar performance had we used SAC.

appendix for more details. Our VAE has a Gaussian decoder with identity variance, meaning that we train the decoder with a mean-squared error loss.

We estimate the density under the VAE by using a sample-wise approximation to the marginal over x estimated using importance sampling:

$$\begin{aligned} p_{\phi_t}(x) &= \mathbb{E}_{z \sim q_{\theta_t}(z|x)} \left[\frac{p(z)}{q_{\theta_t}(z|x)} p_{\psi_t}(x \mid z) \right] \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \left[\frac{p(z)}{q_{\theta_t}(z|x)} p_{\psi_t}(x \mid z) \right]. \end{aligned}$$

where q_θ is the encoder, p_ψ is the decoder, and $p(z)$ is the prior, which in this case is unit Gaussian. In practice we found that sampling $N = 10$ latents for estimating the density to work well in practice.

When training the VAE alongside RL, we found the following two schedules to be effective for different environments:

1. For first $5K$ steps: Train VAE using standard MLE training every 500 time steps for 1000 batches. After that, train VAE using Skew-Fit every 500 time steps for 200 batches.
2. For first $5K$ steps: Train VAE using standard MLE training every 500 time steps for 1000 batches. For the next $45K$ steps, train VAE using Skew-Fit every 500 steps for 200 batches. After that, train VAE using Skew-Fit every 1000 time steps for 200 batches.

We found that initially training the VAE without Skew-Fit improved the stability of the algorithm. This is due to the fact that density estimates under the VAE are extremely unstable and inaccurate during the early phases of training. As a result, we simply train using MLE training at first, and once the density estimates stabilize, we perform Skew-Fit. Table 2 lists the hyper-parameters that were shared across the continuous control experiments. Table 3 lists hyper-parameters specific to each environment.

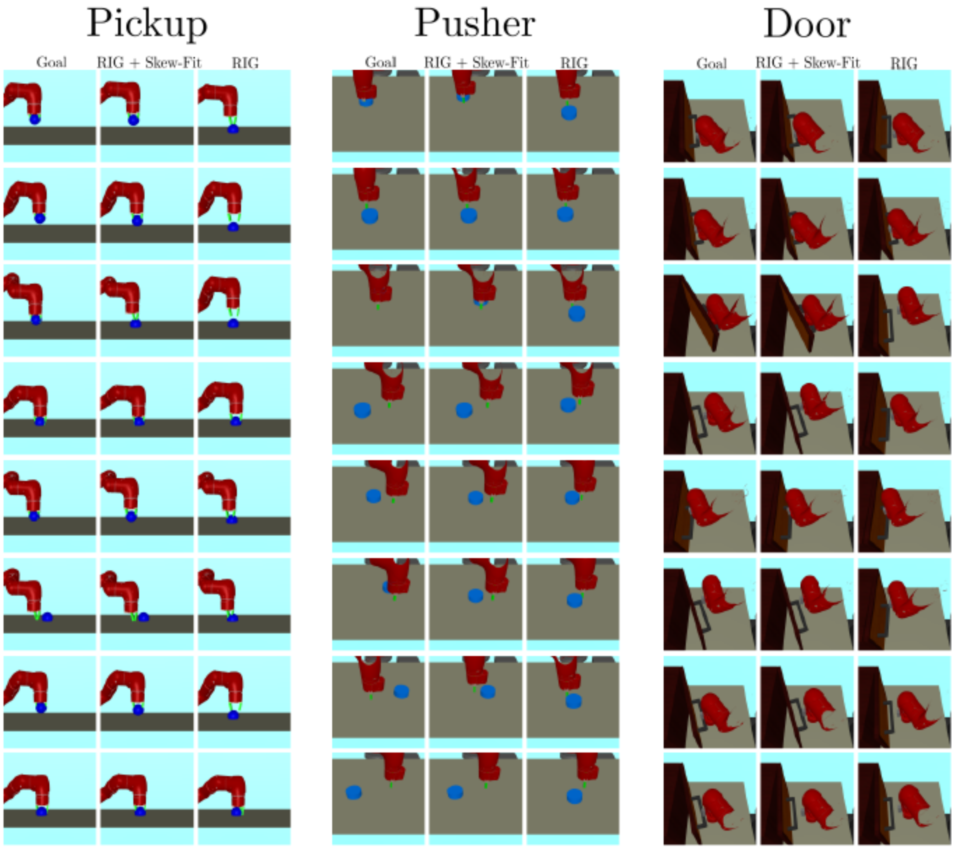


Figure 11: Example reached goals by RIG + Skew-Fit and RIG. The first column of each environment section specifies the target goal while the second and third columns show reached goals by RIG + Skew-Fit and RIG. Both methods learn how to reach goals close to the initial position, but only RIG + Skew-Fit learns to reach the more difficult goals.

Hyper-parameter	**Value**	**Comments**
# training batches per time step	2	Marginal improvements after 2
Exploration Noise	SAC policy	Did not tune
RL Batch Size	1024	smaller batch sizes work as well
VAE Batch Size	64	Did not tune
Discount Factor	0.99	Did not tune
Reward Scaling	1	Did not tune
Path length	100	Did not tune
# of latents for estimating density (N)	10	Marginal improvements after 10

Table 2: General hyper-parameters used for all continuous control experiments.

Hyper-parameter	**Visual Pusher**	**Visual Door**	**Visual Pickup**	**Real World Visual Door**
Path Length	50	100	50	100
β for β-VAE	20	20	30	60
Latent Dimension Size	4	16	16	16
α for Skew-Fit	-1	$-1/2$	-1	$-1/2$
VAE Training Schedule	2	1	2	1
Sample Goals From	p_ϕ	p_{skewed}	p_{skewed}	p_{skewed}

Table 3: Environment specific hyper-parameters

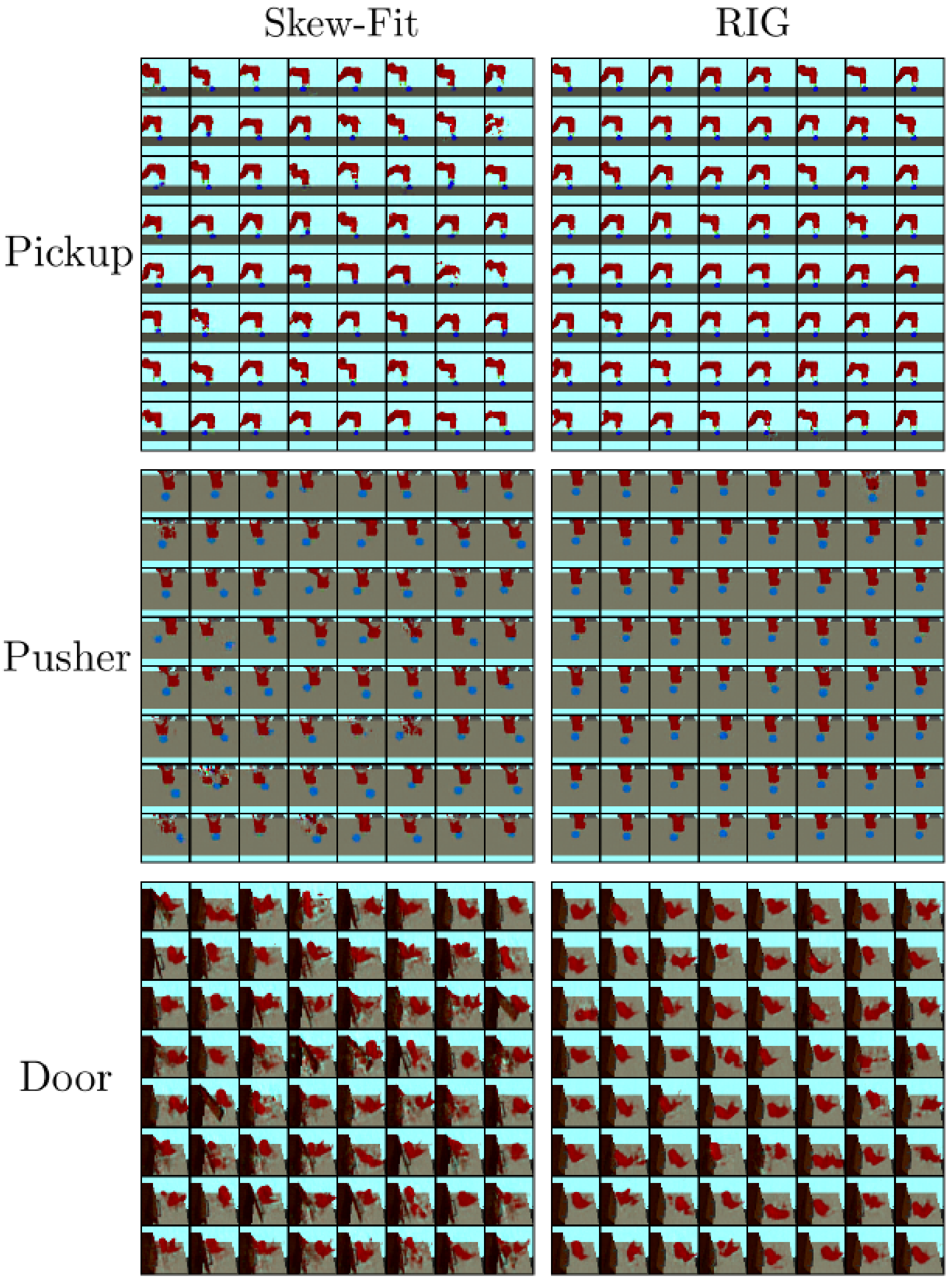


Figure 12: Proposed goals from the VAE for RIG and with RIG + Skew-Fit on the *Visual Pickup*, *Visual Pusher*, and *Visual Door* environments. Standard RIG produces goals where the door is closed and the object and puck is in the same position, while RIG + Skew-Fit proposes goals with varied puck positions, occasional object goals in the air, and both open and closed door angles.

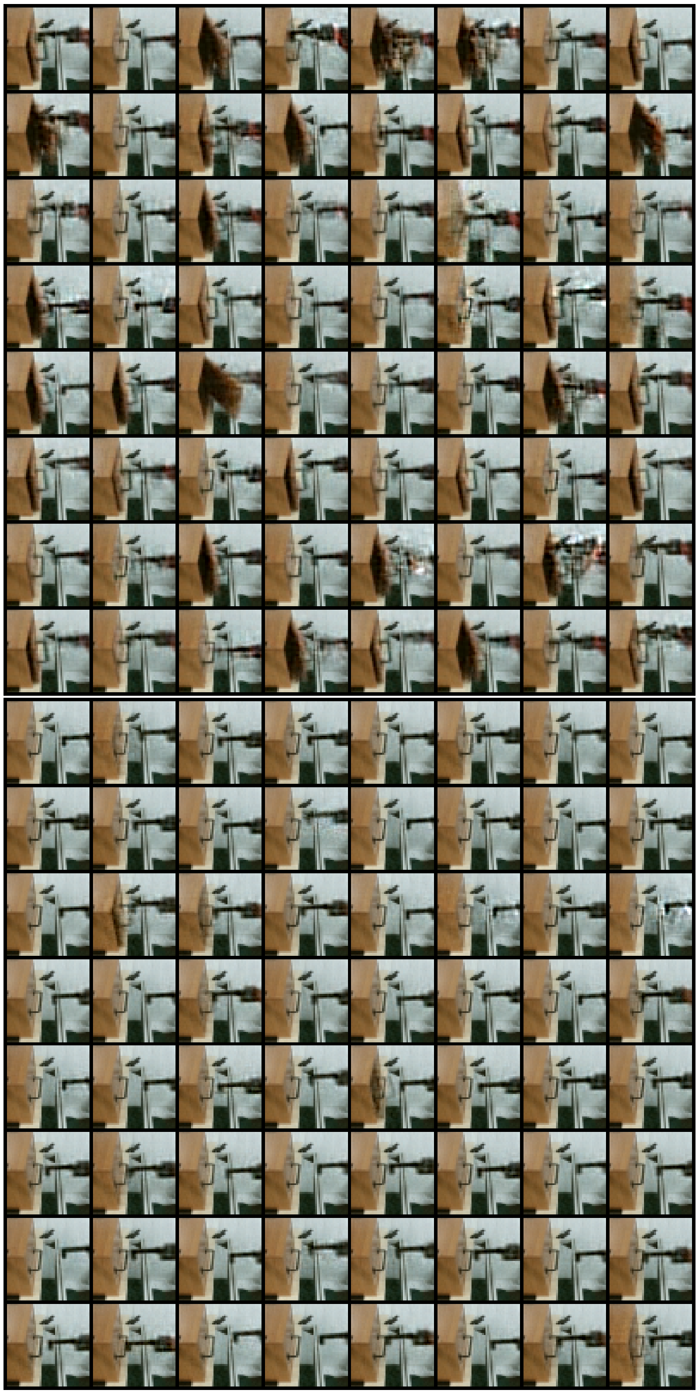


Figure 13: Proposed goals from the VAE for RIG and with RIG + Skew-Fit on the *Visual Pickup*, *Visual Pusher*, and *Visual Door* environments. Standard RIG produces goals where the door is closed and the object and puck is in the same position, while RIG + Skew-Fit proposes goals with varied puck positions, occasional object goals in the air, and both open and closed door angles.