A. Proofs

The definitions of continuity and convergence for pseudo-metrics are similar to those for metrics, and we state them below.

A function $f : \mathcal{Q} \mapsto \mathcal{Q}$ is continuous with respect to a pseudo-metric d if for any $p \in \mathcal{Q}$ and any scalar $\epsilon > 0$, there exists a δ such that for all $q \in \mathcal{Q}$,

$$d(p, q) < \delta \implies d(f(p), f(q)) < \epsilon.$$

An infinite sequence $p_1, p_2 \dots$ converges to a value p with respect to a pseudo-metric d, which we write as

$$\lim_{t\to\infty} p_t \to p,$$

if

$$\lim_{t\to\infty} d(p_t, p) \to 0.$$

Note that if f is continuous, then

$$\lim_{t\to\infty} d_{\mathcal{H}}(p_t, q) \to 0 \implies \lim_{t\to\infty} d_{\mathcal{H}}(f(p_t), f(q)) \to 0.$$

A.1. Proof of Lemma 3.1

Lemma A.1. *Let $\mathcal{S}$ be a compact set. Define the set of distributions $\mathcal{Q} = \{p : \text{support}(p) \subseteq \mathcal{S}\}$. Let $\mathcal{F} : \mathcal{Q} \mapsto \mathcal{Q}$ be continuous with respect to the pseudometric $d_{\mathcal{H}}(p, q) \triangleq |\mathcal{H}(p) - \mathcal{H}(q)|$ and $\mathcal{H}(\mathcal{F}(p)) \geq \mathcal{H}(p)$ with equality if and only if p is the uniform probability distribution on $\mathcal{S}$, denoted as $U_{\mathcal{S}}$. 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 $U_{\mathcal{S}}$ with respect to $d_{\mathcal{H}}$. In other words, $\lim_{t\to 0} |\mathcal{H}(p_t) - \mathcal{H}(U_{\mathcal{S}})| \to 0$.*

Proof. The idea of the proof is to show that the distance (with respect to $d_{\mathcal{H}}$) between p_t and $U_{\mathcal{S}}$ converges to a value. If this value is 0, then the proof is complete since $U_{\mathcal{S}}$ uniquely has zero distance to itself. Otherwise, we will show that this implies that $\mathcal{F}$ is not continuous, which a contradiction.

For shorthand, define d_t to be the $d_{\mathcal{H}}$-distance to the uniform distribution, as in

$$d_t \triangleq d_{\mathcal{H}}(p_t, U_{\mathcal{S}}).$$

First we prove that d_t converges. Since the entropies of the sequence $(p_1, \dots)$ monotonically increase, we have that

$$d_1 \geq d_2 \geq \dots.$$

We also know that d_t is lower bounded by 0, and so by the monotonic convergence theorem, we have that

$$\lim_{t\to\infty} d_t \to d^*.$$

for some value $d^* \geq 0$.

To prove the lemma, we want to show that $d^* = 0$. Suppose, for contradiction, that $d^* \neq 0$. Then consider any distribution, q^*, such that $d_{\mathcal{H}}(q^*, U_{\mathcal{S}}) = d^*$. Such a distribution always exists since we can continuously interpolate entropy values between $\mathcal{H}(p_1)$ and $\mathcal{H}(U_{\mathcal{S}})$ with a mixture distribution. Note that $q^* \neq U_{\mathcal{S}}$ since $d_{\mathcal{H}}(U_{\mathcal{S}}, U_{\mathcal{S}}) = 0$. Since $\lim_{t\to\infty} d_t \to d^*$, we have that

$$\lim_{t\to\infty} d_{\mathcal{H}}(p_t, q^*) \to 0, \tag{5}$$

and so

$$\lim_{t\to\infty} p_t \to q^*.$$

Because the function $\mathcal{F}$ is continuous with respect to $d_{\mathcal{H}}$, Equation 5 implies that

$$\lim_{t\to\infty} d_{\mathcal{H}}(\mathcal{F}(p_t), \mathcal{F}(q^*)) \to 0.$$

However, since $\mathcal{F}(p_t) = p_{t+1}$ we can equivalently write the above equation as

$$\lim_{t\to\infty} d_{\mathcal{H}}(p_{t+1}, \mathcal{F}(q^*)) \to 0.$$

which, through a change of index variables, implies that

$$\lim_{t\to\infty} p_t \to \mathcal{F}(q^*)$$

Since q^* is not the uniform distribution, we have that $\mathcal{H}(\mathcal{F}(q^*)) > \mathcal{H}(q^*)$, which implies that $\mathcal{F}(q^*)$ and q^* are unique distributions. So, p_t converges to two distinct values, q^* and $\mathcal{F}(q^*)$, which is a contradiction. Thus, it must be the case that $d^* = 0$, completing the proof. □

A.2. Proof of Lemma 3.2

Lemma A.2. *Given two distribution $p(x)$ and $q(x)$ where $p \ll q$ and*

$$0 < \text{Cov}_p[\log p(X), \log q(X)] \tag{6}$$

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 $\alpha \in [-a, 0)$,

$$\mathcal{H}_\alpha(\alpha) > \mathcal{H}_\alpha(0) = \mathcal{H}(p). \tag{7}$$

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 $\alpha \in [-a, 0]$, $\mathcal{H}_\alpha(\alpha) > \mathcal{H}_\alpha(0) = \mathcal{H}(p)$. □

The paper applies to the case where $q = q_\phi^G$ and $p = p_\phi^S$. When we take $N \to \infty$, we have that p_{skewed} corresponds to p_α above.

A.3. Simple Case Proof

We prove the convergence directly for the (even more) simplified case when $p_\theta = p(\mathbf{S} \mid q_{\phi_t}^G)$ using a similar technique:

Lemma A.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

$$\begin{aligned}\frac{d}{d\theta} d\mathcal{H}_\theta^t(\theta) &= -\theta A''(\theta) \\ &= -\theta \mathrm{Var}_{\mathbf{s} \sim p_\theta^t}[T(\mathbf{s})] \\ &= -\theta \mathrm{Var}_{\mathbf{s} \sim p_\theta^t}[\log p_t(\mathbf{s})] \\ &\leq 0\end{aligned} \tag{8}$$

where we use the fact that the nth derivative of $A(\theta)$ is the n central moment, i.e. $A''(\theta) = \mathrm{Var}_{\mathbf{s} \sim p_\theta^t}[T(\mathbf{s})]$. 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

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

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 $\frac{d}{d\theta}\mathcal{H}_\theta^t(\theta)$ converges to zero. However, because α is bounded away from 0, Equation 8 states that this can only happen if

$$\mathrm{Var}_{\mathbf{s} \sim p_\theta^t}[\log p_t(\mathbf{s})] \to 0. \tag{9}$$

Because p_t has full support, then so does p_θ^t. Thus, Equation 9 is only true if $\log p_t(\mathbf{s})$ converges to a constant, i.e. p_t converges to the uniform distribution. □

B. Additional Experiments

B.1. Sensitivity Analysis

Sensitivity to RL Algorithm In our experiments, we combined Skew-Fit with soft actor critic (SAC) (Haarnoja et al., 2018). We conduct a set of experiments to test whether Skew-Fit may be used with other RL algorithms for training the goal-conditioned policy. To that end, we replaced

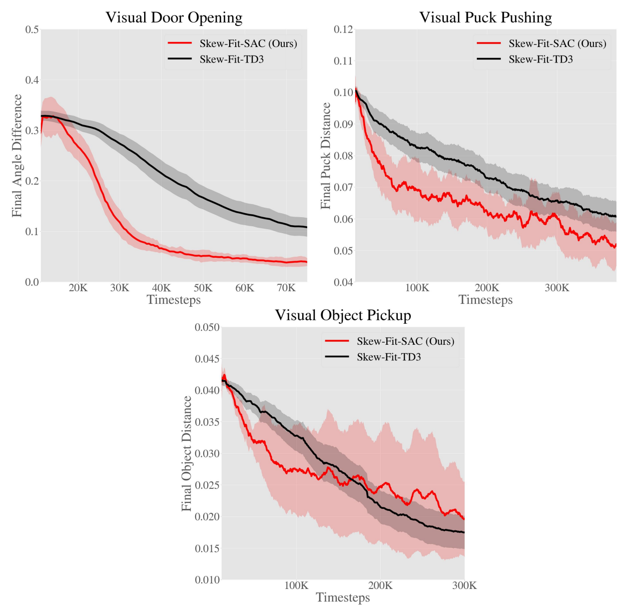

Figure 9. We compare using SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018) as the underlying RL algorithm on Visual Door, Visual Pusher and Visual Pickup. We see that Skew-Fit works consistently well with both SAC and TD3, demonstrating that Skew-Fit may be used with various RL algorithms. For the experiments presented in Section 6, we used SAC.

SAC with twin delayed deep deterministic policy gradient (TD3) (Fujimoto et al., 2018) and ran the same Skew-Fit experiments on Visual Door, Visual Pusher, and Visual Pickup. In Figure 9, we see that Skew-Fit performs consistently well with both SAC and TD3, demonstrating that Skew-Fit is beneficial across multiple RL algorithms.

Sensitivity to α Hyperparameter We study the sensitivity of the α hyperparameter by testing values of $\alpha \in [-1, -0.75, -0.5, -0.25, 0]$ on the Visual Door and Visual Pusher task. The results are included in Figure 10 and shows that our method is robust to different parameters of α, particularly for the more challenging Visual Pusher task. Also, the method consistently outperform $\alpha = 0$, which is equivalent to sampling uniformly from the replay buffer.

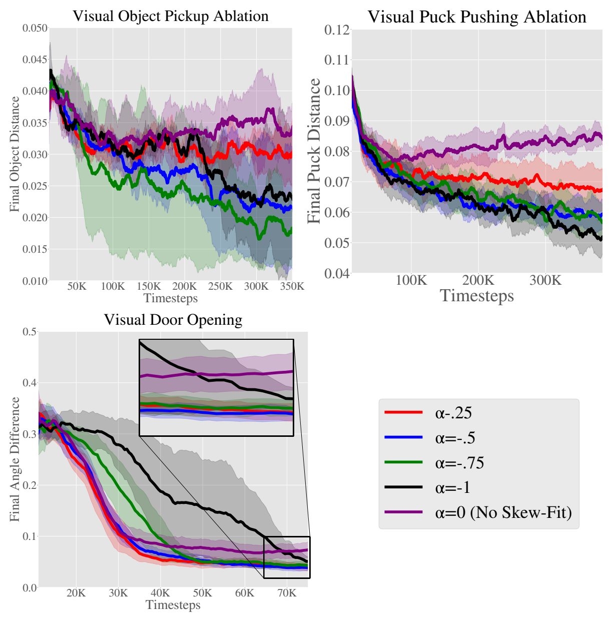

Figure 10. We sweep different values of α on Visual Door, Visual Pusher and Visual Pickup. Skew-Fit helps the final performance on the Visual Door task, and outperforms No Skew-Fit ($\alpha = 0$) as seen in the zoomed in version of the plot. In the more challenging Visual Pusher task, we see that Skew-Fit consistently helps and halves the final distance. Similarly, we observe that Skew-Fit consistently outperforms No Skew-fit on Visual Pickup. Note that alpha=-1 is not always the optimal setting for each environment, but outperforms $\alpha = 0$ in each case in terms of final performance.

B.2. Variance Ablation

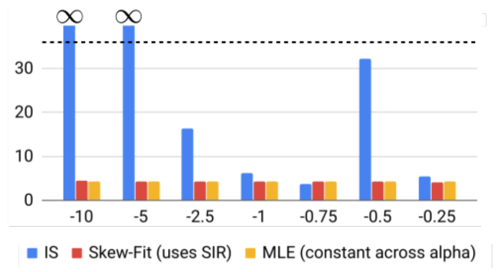

Figure 11. Gradient variance averaged across parameters in last epoch of training VAEs. Values of α less than -1 are numerically unstable for importance sampling (IS), but not for Skew-Fit.

We measure the gradient variance of training a VAE on an unbalanced Visual Door image dataset with Skew-Fit vs Skew-Fit with importance sampling (IS) vs no Skew-Fit (labeled MLE). We construct the imbalanced dataset by rolling out a random policy in the environment and collecting the visual observations. Most of the images contained the door in a closed position; in a few, the door was opened. In Figure 11, we see that the gradient variance for Skew-Fit with IS is catastrophically large for large values of α. In contrast, for Skew-Fit with SIR, which is what we use in practice, the

Method	NLL
MLE on uniform (oracle)	20175.4
Skew-Fit on unbalanced	20175.9
MLE on unbalanced	20178.03

Table 1. Despite training on a unbalanced Visual Door dataset (see Figure 7 of paper), the negative log-likelihood (NLL) of Skew-Fit evaluated on a uniform dataset matches that of a VAE trained on a uniform dataset.

variance is relatively similar to that of MLE. Additionally we trained three VAE's, one with MLE on a uniform dataset of valid door opening images, one with Skew-Fit on the unbalanced dataset from above, and one with MLE on the same unbalanced dataset. As expected, the VAE that has access to the uniform dataset gets the lowest negative log likelihood score. This is the oracle method, since in practice we would only have access to imbalanced data. As shown in Table 1, Skew-Fit considerably outperforms MLE, getting a much closer to oracle log likelihood score.

B.3. Goal and Performance Visualization

We visualize the goals sampled from Skew-Fit as well as those sampled when using the prior method, RIG (Nair et al., 2018). As shown in Figure 12 and Figure 13, the generative model q_ϕ^G results in much more diverse samples when trained with Skew-Fit. We we see in Figure 14, this results in a policy that more consistently reaches the goal image.

C. Implementation Details

C.1. Likelihood Estimation using β-VAE

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

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

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

C.2. Oracle 2D Navigation Experiments

We initialize the VAE to the bottom left corner of the environment for *Four Rooms*. Both the encoder and decoder have 2 hidden layers with [400, 300] units, ReLU hidden activations, and no output activations. The VAE has a latent dimension of 8 and a Gaussian decoder trained with a fixed variance, batch size of 256, and 1000 batches at each iteration. The VAE is trained on the exploration data buffer every 1000 rollouts.

C.3. Implementation of SAC and Prior Work

For all experiments, we trained the goal-conditioned policy using soft actor critic (SAC) (Haarnoja et al., 2018). To make the method goal-conditioned, we concatenate the target XY-goal to the state vector. During training, we retroactively relabel the goals (Kaelbling, 1993; Andrychowicz et al., 2017) by sampling from the goal distribution with probabilty 0.5. Note that the original RIG (Nair et al., 2018) paper used TD3 (Fujimoto et al., 2018), which we also replaced with SAC in our implementation of RIG. 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. In the prior RIG method, the VAE was pre-trained on a uniform sampling of images from the state space of each environment. In order to ensure a fair comparison to Skew-Fit, we forego pre-training and instead train the VAE alongside RL, using the variant described in the RIG paper. 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 continuously train the VAE concurrently with RL training). 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.

For our implementation of (Hazan et al., 2019), we trained the policies with the reward

$$r(s) = r_{\text{Skew-Fit}}(s) + \lambda \cdot r_{\text{Hazan et al.}}(s)$$

For $r_{\text{Hazan et al.}}$, we use the reward described in Section 5.2 of Hazan et al. (2019), which requires an estimated likelihood of the state. To compute these likelihood, we use the same method as in Skew-Fit (see Appendix C.1). With 3 seeds each, we tuned λ across values $[100, 10, 1, 0.1, 0.01, 0.001]$ for the door task, but all values performed poorly. For the pushing and picking tasks, we tested values across $[1, 0.1, 0.01, 0.001, 0.0001]$ and found that 0.1 and 0.01 performed best for each task, respectively.

C.4. RIG with Skew-Fit Summary

Algorithm 2 provides detailed pseudo-code for how we combined our method with RIG. Steps that were removed from the base RIG algorithm are highlighted in blue and

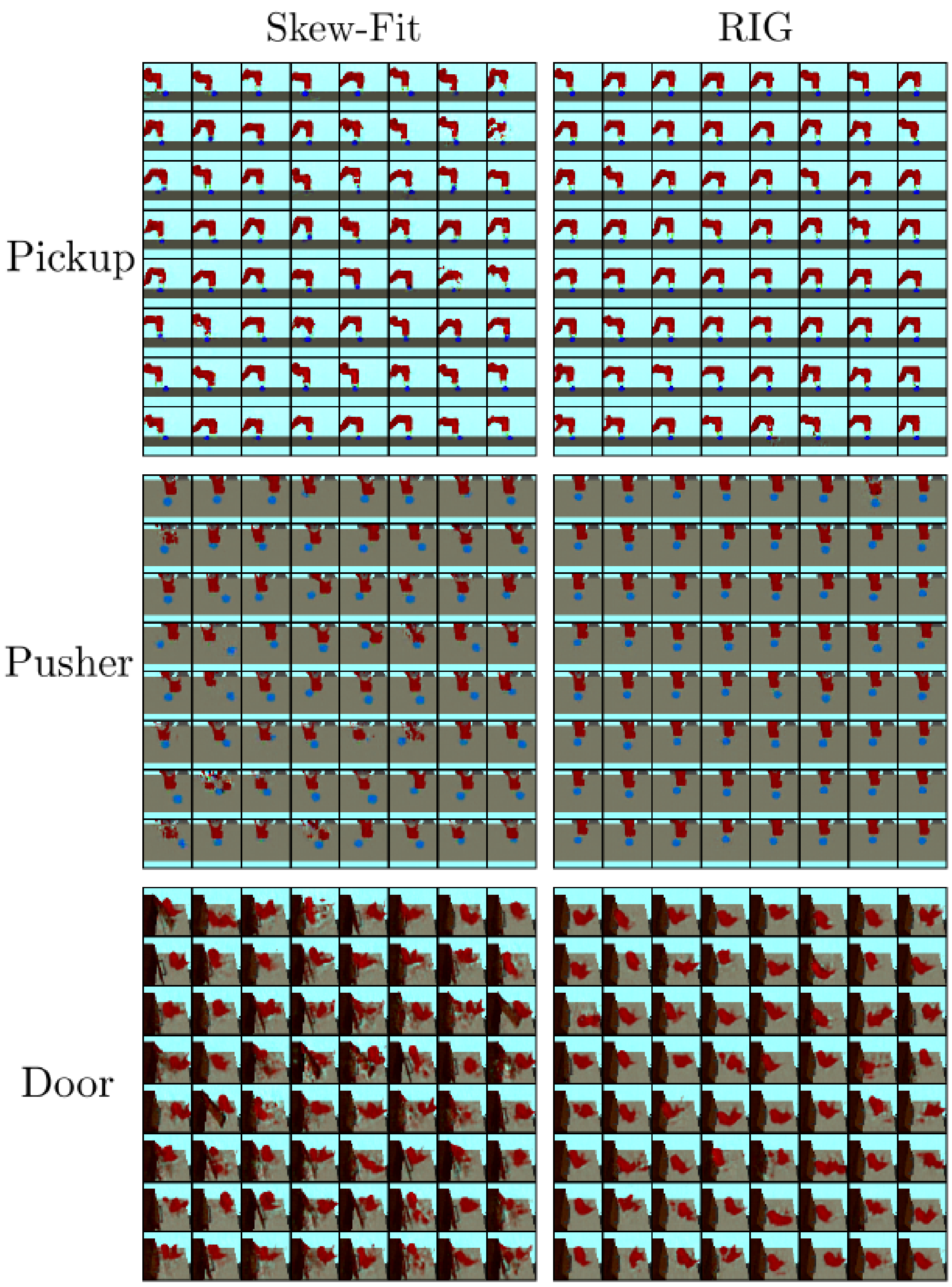

Figure 12. Proposed goals from the VAE for RIG and with Skew-Fit on the *Visual Pickup*, *Visual Pusher*, and *Visual Door* environments. Standard RIG produces goals where the door is closed and the object and puck is in the same position, while RIG + Skew-Fit proposes goals with varied puck positions,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

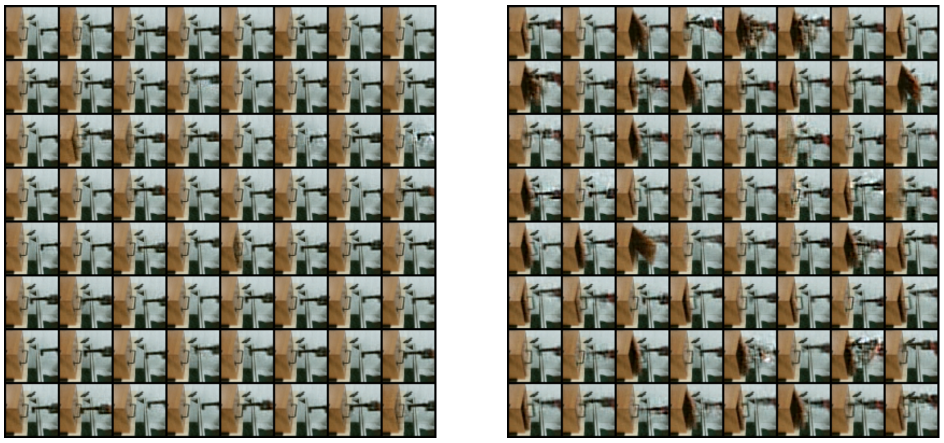

Figure 13. Proposed goals from the VAE for RIG (left) and with RIG + Skew-Fit (right) on the *Real World Visual Door* environment. Standard RIG produces goals where the door is closed while RIG + Skew-Fit proposes goals with both open and closed door angles.

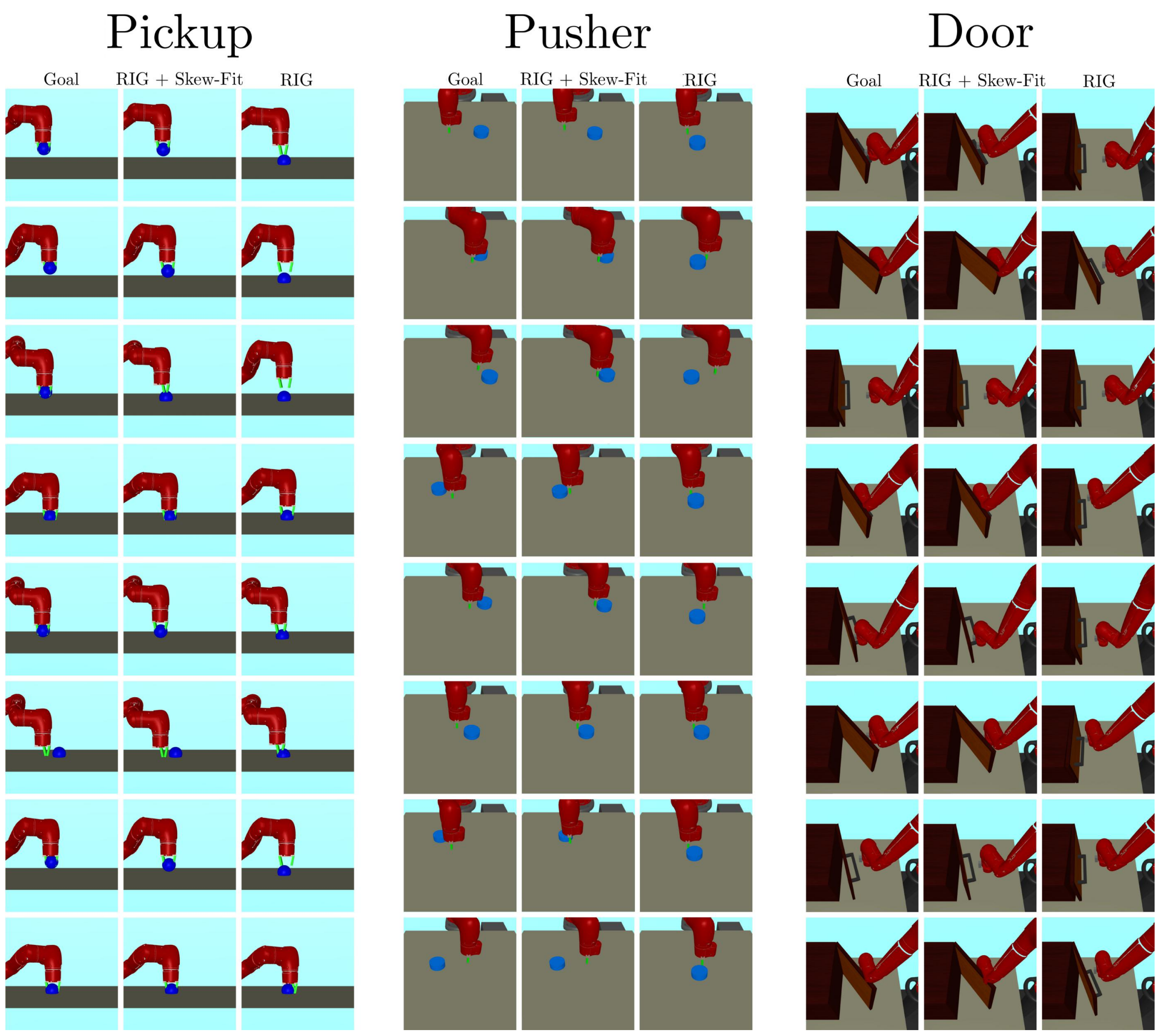

Figure 14. Example reached goals by Skew-Fit and RIG. The first column of each environment section specifies the target goal while the second and third columns show reached goals by Skew-Fit and RIG. Both methods learn how to reach goals close to the initial position, but only Skew-Fit learns to reach the more difficult goals.

steps that were added are highlighted in red. The main differences between the two are (1) not needing to pre-train the β-VAE, (2) sampling exploration goals from the buffer using p_{skewed} instead of the VAE prior, (3) relabeling with replay buffer goals sampled using p_{skewed} instead of from the VAE prior, and (4) training the VAE on replay buffer data data sampled using p_{skewed} instead of uniformly.

C.5. Vision-Based Continuous Control Experiments

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 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.

When training the VAE alongside RL, we found the following three 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.

3. For first $40K$ steps: Train VAE using standard MLE training every 4000 time steps for 1000 batches. Afterwards, train VAE using Skew-Fit every 4000 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 constantly changing and inaccurate during the early phases of training. Therefore, it made little sense to use those estimates to prioritize goals early on in training. Instead, we simply train using MLE training for the first $5K$ timesteps, and after that we perform Skew-Fit according to the VAE schedules above. Table 2 lists the hyper-parameters that were shared across the continuous control experiments. Table 3 lists hyper-parameters specific to each environment. Additionally, Appendix C.4 discusses the combined RIG + Skew-Fit algorithm.

Algorithm 2 RIG and RIG + Skew-Fit. Blue text denotes RIG specific steps and red text denotes RIG + Skew-Fit specific steps

```
Require: β-VAE mean encoder $q_\phi$, β-VAE decoder $p_\psi$, policy $\pi_\theta$, goal-conditioned value function $Q_w$, skew parameter $\alpha$, VAE Training Schedule.
1: Collect $\mathcal{D} = \{s^{(i)}\}$ using random initial policy.
2: Train $\beta$-VAE on data uniformly sampled from $\mathcal{D}$.
3: Fit prior $p(z)$ to latent encodings $\{\mu_\phi(s^{(i)})\}$.
4: for $n = 0, ..., N-1$ episodes do
5:   Sample latent goal from prior $z_g \sim p(z)$.
6:   Sample state $s_g \sim p_{\text{skewed}_n}$ and encode $z_g = q_\phi(s_g)$ if $\mathcal{R}$ is nonempty. Otherwise sample $z_g \sim p(z)$
7:   Sample initial state $s_0$ from the environment.
8:   for $t = 0, ..., H-1$ steps do
9:     Get action $a_t \sim \pi_\theta(q_\phi(s_t), z_g)$.
10:    Get next state $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.
11:    Store $(s_t, a_t, s_{t+1}, z_g)$ into replay buffer $\mathcal{R}$.
12:    Sample transition $(s, a, s', z_g) \sim \mathcal{R}$.
13:    Encode $z = q_\phi(s), z' = q_\phi(s')$.
14:    (Probability 0.5) replace $z_g$ with $z'_g \sim p(z)$.
15:    (Probability 0.5) replace $z_g$ with $q_\phi(s'')$ where $s'' \sim p_{\text{skewed}_n}$.
16:    Compute new reward $r = -||z' - z_g||$.
17:    Minimize Bellman Error using $(z, a, z', z_g, r)$.
18:  end for
19:  for $t = 0, ..., H-1$ steps do
20:    for $i = 0, ..., k-1$ steps do
21:      Sample future state $s_{h_i}, t < h_i \leq H-1$.
22:      Store $(s_t, a_t, s_{t+1}, q_\phi(s_{h_i}))$ into $\mathcal{R}$.
23:    end for
24:  end for
25:  Construct skewed replay buffer distribution $p_{\text{skewed}_{n+1}}$ using data from $\mathcal{R}$ with Equation 3.
26:  if total steps $< 5000$ then
27:    Fine-tune $\beta$-VAE on data uniformly sampled from $\mathcal{R}$ according to VAE Training Schedule.
28:  else
29:    Fine-tune $\beta$-VAE on data uniformly sampled from $\mathcal{R}$ according to VAE Training Schedule.
30:    Fine-tune $\beta$-VAE on data sampled from $p_{\text{skewed}_{n+1}}$ according to VAE Training Schedule.
31:  end if
32: end for
```

D. Environment Details

Four Rooms: A 20 x 20 2D pointmass environment in the shape of four rooms (Sutton et al., 1999). The observation is the 2D position of the agent, and the agent must specify a target 2D position as the action. The dynamics of the environment are the following: first, the agent is teleported to the target position, specified by the action. Then a Gaussian change in position with mean 0 and standard deviation 0.0605 is applied[6]. If the action would result in the agent

[6]In the main paper, we rounded this to 0.06, but this difference does not matter.

Hyper-parameter	Value	Comments
# training batches per time step	2	Marginal improvements after 2
Exploration Noise	None (SAC policy is stochastic)	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
Policy Hidden Sizes	$[400, 300]$	Did not tune
Policy Hidden Activation	ReLU	Did not tune
Q-Function Hidden Sizes	$[400, 300]$	Did not tune
Q-Function Hidden Activation	ReLU	Did not tune
Replay Buffer Size	100000	Did not tune
Number of Latents for Estimating Density (N)	10	Marginal improvements beyond 10

Table 2. General hyper-parameters used for all *visual* 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	q_ϕ^G	p_{skewed}	p_{skewed}	p_{skewed}

Table 3. Environment specific hyper-parameters for the *visual* experiments

Hyper-parameter	Value
# training batches per time step	.25
Exploration Noise	None (SAC policy is stochastic)
RL Batch Size	512
VAE Batch Size	64
Discount Factor	$\frac{299}{300}$
Reward Scaling	10
Path length	300
Policy Hidden Sizes	$[400, 300]$
Policy Hidden Activation	ReLU
Q-Function Hidden Sizes	$[400, 300]$
Q-Function Hidden Activation	ReLU
Replay Buffer Size	1000000
Number of Latents for Estimating Density (N)	10
β for β-VAE	10
Latent Dimension Size	2
α for Skew-Fit	-2.5
VAE Training Schedule	3
Sample Goals From	p_{skewed}

Table 4. Hyper-parameters used for the *ant* experiment.

moving through or into a wall, then the agent will be stopped at the wall instead.

Ant: A MuJoCo (Todorov et al., 2012) ant environment. The observation is a 3D position and velocities, orientation, joint angles, and velocity of the joint angles of the ant (8 total). The observation space is 29 dimensions. The agent controls the ant through the joints, which is 8 dimensions. The goal is a target 2D position, and the reward is the negative Euclidean distance between the achieved 2D position and target 2D 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. 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 $[0, 45]$ degrees.

E. Goal-Conditioned Reinforcement Learning Minimizes $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$

Some goal-conditioned RL methods such as Warde-Farley et al. (2018); Nair et al. (2018) present methods for minimizing a lower bound for $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$, by approximating $\log p(\mathbf{G} \mid \mathbf{S})$ and using it as the reward. Other goal-conditioned RL methods (Kaelbling, 1993; Lillicrap et al., 2016; Schaul et al., 2015; Andrychowicz et al., 2017; Pong et al., 2018; Florensa et al., 2018a) are not developed with the intention of minimizing the conditional entropy $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$. Nevertheless, one can see that goal-conditioned RL generally minimizes $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$ by noting that the optimal goal-conditioned policy will deterministically reach the goal. The corresponding conditional entropy of the goal given the state, $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$, would be zero, since given the current state, there would be no uncertainty over the goal (the goal must have been the current state since the policy is optimal). So, the objective of goal-conditioned RL can be interpreted as finding a policy such that $\mathcal{H}(\mathbf{G} \mid \mathbf{S}) = 0$. Since zero is the minimum value of $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$, then goal-conditioned RL can be interpreted as minimizing $\mathcal{H}(\mathbf{G} \mid \mathbf{S})$.