Hindsight Expectation Maximization for Goal-conditioned Reinforcement Learning

Yunhao Tang  
Columbia University  
yt2541@columbia.edu

Alp Kucukelbir  
Columbia University  
alp@cs.columbia.edu

Abstract

We propose a graphical model framework for goal-conditioned reinforcement learning (RL), with an expectation maximization (EM) algorithm that operates on the lower bound of the RL objective. The E-step provides a natural interpretation of how ‘learning in hindsight’ techniques, such as hindsight experience replay (HER), to handle extremely sparse goal-conditioned rewards. The M-step reduces policy optimization to supervised learning updates, which greatly stabilizes end-to-end training on high-dimensional inputs such as images. We show that the combined algorithm, hindsight expectation maximization (hEM) significantly outperforms model-free baselines on a wide range of goal-conditioned benchmarks with sparse rewards.

1 Introduction

In goal-conditioned reinforcement learning (RL), an agent seeks to achieve a goal through interactions with the environment. At each step, the agent receives a reward, which ideally reflects how well it is achieving its goal. Traditional RL methods leverage these rewards to learn good policies. As such, the effectiveness of these methods rely on how informative the rewards are.

This sensitivity of traditional RL algorithms has led to a flurry of activity around reward shaping [1]. This limits the applicability of RL, as reward shaping is often specific to an environment and task — a practical obstacle to wider applicability. Binary rewards, however, are trivial to specify. The agent receives a strict indicator of success when it has achieved its goal; until then, it receives precisely zero reward. Such a sparsity of reward signals renders goal-conditioned RL extremely challenging for traditional methods [2].

How can we navigate such binary reward settings? Consider an agent that explores its environment but fails to achieve its goal. One idea is to treat, in hindsight, its exploration as having achieved some other goal. By relabeling a ‘failure’ relative to an original goal as a ‘success’ with respect to some other goal, we can imagine an agent succeeding frequently at many goals, in spite of failing at the original goals. This insight motivates hindsight experience replay (HER) [2], an intuitive strategy that enables off-policy RL algorithms, such as [3, 4], to function in sparse binary reward settings.

The statistical simulation of rare events occupies a similar setting. Consider estimating an expectation of low-probability events using Monte Carlo sampling. The variance of this estimator relative to its expectation is too high to be practical [5]. A powerful approach to reduce variance is importance sampling (IS) [6]. The idea is to adapt the sampling procedure such that these rare events occur frequently, and then to adjust the final computation. Could IS help in binary reward RL settings too?

Main idea. We propose a probabilistic framework for goal-conditioned RL that formalizes the intuition of hindsight using ideas from statistical simulation. We equate the traditional RL objective to maximizing the evidence of our probabilistic model. This leads to a new algorithm, hindsight
expectation maximization (hEM), which maximizes a tractable lower bound of the original objective [7]. A central insight is that the E-step naturally interprets hindsight replay as a special case of IS.

Figure 1 compares hEM to HER [2] on four goal-conditioned RL tasks with low-dimensional state and high-dimensional images as inputs. While hEM performs consistently well on both inputs, HER struggles with image-based inputs. This is due to how HER leverages hindsight experiences through the lens of IS, thus enabling better performance in high dimensions.

The rest of this section presents a quick background on goal-conditioned RL and probabilistic inference. Expert readers may jump ahead to Section 2, which presents our graphical model.

**Goal-conditioned RL background.** A Markov decision process (MDP) can be simply extended to incorporate multiple goals. Consider an agent that interacts with an environment in episodes. At the beginning of each episode, a goal $g \in G$ is fixed. At a discrete time $t \geq 0$, an agent in state $s_t \in S$ takes action $a_t \in A$ receives a reward $r(s_t, a_t, g) \in \mathbb{R}$ and transitions to its next state $s_{t+1} \sim p(\cdot | s_t, a_t) \in S$. This process is independent of goals. A policy $\pi(a | s, g) : S \times G \to P(A)$ defines a map from state and goal to distributions over actions. Given a distribution over goals $g \sim p(\cdot)$, we consider the undiscounted episodic return $J(\pi) := \mathbb{E}_{g \sim p(\cdot)} \left[ \mathbb{E}_z \left[ \sum_{t=0}^{T-1} r(s_t, a_t, g) \right] \right]$. When rewards are independent from goals $r(s_t, a_t, g) \equiv r(s_t, a_t)$ and we recover classical RL [10].

**Probabilistic inference background.** Consider data as observed random variables $x \in \mathcal{X}$. Each measurement $x$ is a discrete or continuous random variable. A likelihood $p_\theta(x | z)$ relates each measurement to latent variables $z \in \mathcal{Z}$ and unknown, but fixed, parameters $\theta$. The full probabilistic generative model specifies a prior over the latent variable $p(z)$. Bayesian inference requires computing the posterior $p(z | x)$ — an intractable task for all but a small class of simple models.

Variational inference approximates the posterior by matching a tractable density $q_\phi(z | x)$ to the posterior. The following calculation specifies this procedure:

\[
\log p(x) = \log \mathbb{E}_{z \sim p(\cdot)} \left[ p_\theta(x | z) \right] \\
= \log \mathbb{E}_{z \sim q_\phi(\cdot | x)} \left[ p_\theta(x | z) \frac{p(z)}{q_\phi(z | x)} \right] \geq \mathbb{E}_{z \sim q_\phi(\cdot | x)} \left[ \log \frac{p_\theta(x | z)}{q_\phi(z | x)} \right] (1) \\
= \mathbb{E}_{z \sim q_\phi(\cdot | x)} \left[ \log p_\theta(x | z) \right] - \text{KL}(q_\phi(\cdot | z) \parallel p(z)) := L(p_\theta, q), (2)
\]

(For a detailed derivation, please see [7].) Equation (3) defines the evidence lower bound (ELBO) $L(p_\theta, q)$. Matching the tractable density $q_\phi(z | x)$ to the posterior thus turns into maximizing the ELBO via expectation maximization (EM) [11] or stochastic gradient ascent [12, 13]. Figure 2(a) presents a graphical model of the above. For a fixed set of $\theta$ parameters, the optimal variational distribution $q$ is the true posterior $\arg \max_q L(p_\theta, q) \equiv p(z | x) := p(z)p_\theta(x)p(x)$.

From an IS perspective, note how the variational distribution $q_\phi(z | x)$ serves as a proposal distribution in place of $p(z)$.
Appendix A presents a detailed comparison to RL. The following proposition shows the equivalence between inference in this model and traditional Variational RL as inference.

Two recent frameworks connect probabilistic inference to general RL: Variational RL [16–19] and RL as inference [20–22]. We situate our probabilistic model by first presenting Variational RL below. Appendix A presents a detailed comparison to RL as inference.

Variational RL. Begin by defining a trajectory random variable \( \tau \equiv (s_t, a_t)_{t=0}^{T-1} \) to encapsulate a sequence of state and action pairs. The random variable is generated by a factorized distribution \( a_t \sim \pi_\theta(\cdot \mid s_t), s_{t+1} \sim p(\cdot \mid s_t, a_t) \), which defines the joint distribution \( p_\theta(\tau) := \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t) \). Conditional on \( \tau \), define the distribution of a binary optimality variable as \( p(O = 1 \mid \tau) \propto \exp(\sum_{t=0}^{T-1} r(s_t, a_t)/\alpha) \) for some \( \alpha > 0 \), where we assume \( r(s_t, a_t) \geq 0 \) without loss of generality. Optimizing the standard RL objective corresponds to maximizing the evidence of \( \log p(O = 1) \), where all binary variables are treated as observed and equal to one. Positing a variational approximation to the posterior over trajectories gives the following lower bound,

\[
\log p(O = 1) \geq E_{q(\tau)}[\log p(O = 1 \mid \tau)] - KL[q(\tau) \parallel p(\tau)] := L(\pi_\theta, q). \tag{4}
\]

Figure 2(b) shows a combined graphical model for both the generative and inference models of Variational RL. Equation (4) is typically maximized using EM (e.g., [23, 18, 19]), by alternating updates between \( \theta \) and \( q(\tau) \). Note that Variational RL does not model goals.

### 2.1 Probabilistic goal-conditioned reinforcement learning

To extend the Variational RL framework to incorporate goals, introduce a goal variable \( g \) and a prior distribution \( g \sim p(\cdot) \). Conditional on a goal \( g \), the trajectory variable \( \tau \equiv (s_t, a_t)_{t=0}^{T-1} \sim p(\cdot \mid \theta, g) \) is sampled by executing the policy \( \pi_\theta(a_t \mid s_t) \) in the MDP. Similar to Variational RL, the joint distribution factorizes as \( p(\tau \mid \theta, g) := \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t, g)p(s_{t+1} \mid s_t, a_t) \). Now, define a goal-conditioned binary optimality variable \( O \), such that \( p(O = 1 \mid \tau, g) := R(\tau, g) := \sum_{t=0}^{T-1} r(s_t, a_t, g)/\alpha \) where \( \alpha > 0 \) normalizes this density. Figure 2(c) shows a graphical model of just this generative model. Treat the optimality variables as the observations and assume \( O \equiv 1 \). The following proposition shows the equivalence between inference in this model and traditional goal-conditioned RL.

**Proposition 1.** (Proof in Appendix B.) Maximizing the evidence of the probabilistic model is equivalent to maximizing returns in the goal-conditioned RL problem, i.e.,

\[
\arg \max_\theta \log p(O = 1) = \arg \max_\theta J(\pi_\theta). \tag{5}
\]

### 2.2 Challenges with direct optimization

Equation (5) implies that algorithms to maximize the evidence of such probabilistic models could be readily applied to goal-conditioned RL. Unlike typical probabilistic inference settings, the evidence here could technically be directly optimized. Indeed, \( p(O = 1) \equiv J(\pi_\theta) \) could be maximized via traditional RL approaches e.g., policy gradients [24]. In particular, the REINFORCE gradient estimator [25] of Equation (5) is given by as \( \eta_\theta = \sum_{t \geq 0} \sum_{a' \geq 1} r(s_t, a', g) \nabla_\theta \log \pi_\theta(a_t \mid s_t, g) \approx \nabla_\theta J(\pi_\theta) \),
where \( g \sim p(\cdot) \) and \((s_t, a_t)_{t=0}^{T-1}\) are sampled on-policy. The direct optimization of \( \log p(O = 1) \) consists in gradient ascents \( \theta \leftarrow \theta + \eta_\theta \). This poses a practical challenge in goal-conditioned RL. To see why, consider the following example.

**Illustrative example.** Consider a one-step MDP with \( T = 1 \) where \( S = \{s_0\}, A = \mathcal{G}, r(s, a, g) = \mathbb{I}[a = g] \). Assume that there are a finite number of actions and goals \(|A| = |\mathcal{G}| = k\).

The following theorem shows the difficulty in building a practical estimator for \( \nabla_\theta \mathcal{L} \).

**Theorem 1.** (Proof in Appendix B.2.) Consider the example above. Let the policy \( \pi_\theta(a \mid s, g) = \text{softmax}(L_{a,g}) \) be parameterized by logits \( L_{a,g} \) and let \( q_{\theta,g} \) be the one-sample REINFORCE gradient estimator of \( L_{a,g} \). Assume a uniform distribution over goals \( p(g) = 1/k \) for all \( g \in \mathcal{G} \). Assume that the policy is randomly initialized (e.g. \( L_{a,g} \equiv L, \forall a, g \) for some \( L \)). Let \( \mathcal{L} \) be the mean squared error \( \mathbb{E}[x] := \mathbb{E}[(x - \mathbb{E}[p(a \mid \theta, g)])^2] \). It can be shown that the relative error of the estimate \( \sqrt{\mathbb{V}[q_{\theta,g}] / \mathbb{E}[q_{\theta,g}]} = k(1 + o(1)) \) grows approximately linearly with \( k \) for all \( \forall a \in A, g \in \mathcal{G} \).

The above theorem shows that in the simple setup above, the relative error of the REINFORCE gradient estimator grows linearly in \( k \). This implies that to reduce the error with traditional Monte Carlo sampling would require \( m \approx k^2 \) samples, which quickly becomes intractable as \( k \) increases. Though variance reduction methods such as control variates [26] could be of help, it does not change the sup-linear growth rate of samples (see comments in Appendix B.2). The fundamental bottleneck is that dense gradients \( r(s, a, g) \nabla_\theta \log p_\theta(a \mid s, g) \neq 0 \) are rare event with a probability of \( 1/k \), which makes it difficult to accurately estimate with on-policy measures [5]. This example hints at similar issues with more realistic cases and motivates an IS approach to address the problem.

### 2.3 Tractable lower bound

Consider a variational inequality similar to Equation (3) with a variational distribution \( q(\tau, g) \)

\[
\log p(O = 1) = \log \mathbb{E}_{q(\tau, g)} \left[ p(O = 1 \mid \tau, g) \frac{p(g)p(\tau \mid \theta, g)}{q(\tau, g)} \right] 
\geq \mathbb{E}_{q(\tau, g)} \left[ \log p(O = 1 \mid \tau, g) \frac{p(g)p(\tau \mid \theta, g)}{q(\tau, g)} \right] 
= \mathbb{E}_{q(\tau, g)} \left[ \log p(O = 1 \mid \tau, g) - \mathbb{KL}[q(\tau, g) \parallel p(\tau \mid \theta, g)] \right] =: \mathcal{L}(\pi_\theta, q). \tag{8}
\]

This variational distribution corresponds to the inference model in Figure 2(d). As with typical graphical models, instead of maximizing \( \log p(O = 1) \), consider maximizing its ELBO \( \mathcal{L}(\pi_\theta, q) \) with respect to both \( \theta \) and variational distribution \( q(\tau, g) \). Our key insight lies in the following observation: the bottleneck of the direct optimization of \( \log p(O = 1) \) lies in the sparsity of \( p(O = 1 \mid \tau, g) = \sum_{t=0}^{T-1} r(s_t, a_t, g) \), where \( (\tau, g) \) are sampled with the on-policy measure \( g \sim p(\cdot), \tau \sim p(\cdot \mid \theta, g) \).

The variational distribution \( q(\tau, g) \) serves as a IS proposal in place of \( p(\tau \mid \theta, g)p(g) \). If \( q(\tau, g) \) puts more probability mass on \( (\tau, g) \) pairs with high returns (high \( p(O = 1 \mid \tau, g) \)), the rewards become dense and learning becomes feasible. In the next section, we show how hindsight replay [2] provides an intuitive and effective way to select such a \( q(\tau, g) \).

### 3 Hindsight Expectation Maximization

The EM-algorithm [11] for Equation (8) alternates between an E- and M-step: at iteration \( t \), denote the policy parameter to be \( \theta_t \) and the variational distribution to be \( q_t \).

**E-step:** \( q_{t+1} = \arg \max_q \mathcal{L}(\pi_{\theta_t}, q) \), **M-step:** \( \theta_{t+1} = \arg \max_{\theta} \mathcal{L}(\pi_\theta, q_{t+1}) \).

\[
\text{E-step: } q_{t+1} = \arg \max_q \mathcal{L}(\pi_{\theta_t}, q), \quad \text{M-step: } \theta_{t+1} = \arg \max_{\theta} \mathcal{L}(\pi_\theta, q_{t+1}). \tag{9}
\]

This ensures a monotonic improvement in the ELBO \( \mathcal{L}(\pi_{\theta_{t+1}}, q_{t+1}) \geq \mathcal{L}(\pi_{\theta_t}, q_t) \). We discuss these two alternating steps in details below, starting with the M-step.

**M-step: Optimization for \( \pi_\theta \).** Fixing the variational distribution \( q(\tau, g) \), to optimize \( \mathcal{L}(\pi_\theta, q) \) with respect to \( \theta \) is equivalent to

\[
\max_{\theta} \mathbb{E}_{q(\tau, g)} \left[ \log p(\tau \mid \theta, g) \right] \equiv \max_{\theta} \mathbb{E}_{q(\tau, g)} \left[ \sum_{t=0}^{T-1} \log \pi_\theta(a_t \mid s_t, g) \right]. \tag{10}
\]
The right hand side of Equation (10) corresponds to a supervised learning problem where learning samples come from \( q(\tau, g) \). Prior studies have adopted this idea and developed policy optimization algorithms in this direction \([18, 19, 27-29]\). In practice, the M-step is carried out partially where \( \theta \) is updated with gradient steps instead of optimizing Equation (10) fully.

**E-step: Optimization for \( q(\tau, g) \).** The choice of \( q(\tau, g) \) should satisfy two desirable properties:\n\( (P.1) \) it leads to monotonic improvements in \( \log p(O = 1) \equiv J(\pi_\theta) \) or a lower bound thereof; \( (P.2) \) it provide dense learning signals for the M-step. The posterior distribution \( p(\tau, g \mid O = 1) \) achieves \( (P.1) \) and \( (P.2) \) in a near-optimal way, in that it is the maximizer of the E-step in Equation (9), which monotonically improves the ELBO. The posterior also provides dense reward signals to the M-step because \( p(\tau, g \mid O = 1) \propto p(O = 1 \mid \tau, g) \). In practice, one chooses a variational distribution \( q(\tau, g) \) as an alternative to the intractable posterior by maximizing Equation (8). Below, we show it is possible to achieve \( (P.1)(P.2) \) even though the E-step is not carried out fully. By plugging in \( p(O = 1 \mid \tau, g) = \sum_{t=0}^{T-1} r(s_t, a_t, g) / \alpha \), we write the ELBO as

\[
L(\pi_\theta, q) = \mathbb{E}_{q(\tau, g)} \left[ \sum_{t=0}^{T-1} \frac{r(s_t, a_t, g)}{\alpha} \right] + - \text{KL}[q(\tau, g) \parallel p(\tau \mid \theta, g)].
\]  

We now examine alternative ways to select the variational distribution \( q(\tau, g) \).

**Prior work.** State-of-the-art model-free algorithms such as MPO \([18, 19]\) applies a factorized variational distribution \( q_{\text{ent}}(\tau, g) = p(g) \prod_{t=0}^{T-1} q_{\text{ent}}(a_t \mid s_t, g) \). The variational distribution is defined by local distributions \( q_{\text{ent}}(a \mid s, g) : = \pi(\alpha \mid s, g) \exp(\tilde{Q}^\pi(s, a, g) / \eta) \) for some temperature \( \eta > 0 \) and estimates of Q-functions \( \tilde{Q}^\pi(s, a, g) \). The design of \( q_{\text{ent}}(\tau, g) \) could be interpreted as initializing \( q_{\text{ent}}(a \mid s, g) \) with \( \pi(\alpha \mid s, g) \) which effectively maximizes the second term in Equation (11), then taking one improvement step of the first term \([18]\). This distribution satisfies \( (P.1) \) because the combined EM-algorithm corresponds to entropy-regularized policy iteration, and retains monotonic improvements in \( J(\pi_\theta) \). However, it does not satisfy \( (P.2) \): when rewards are sparse \( r(s, a, g) \approx 0 \), estimates of Q-functions are sparse \( Q^\pi(s, a, g) \approx 0 \) and leads to uninformative variational distributions \( q_{\text{ent}}(a \mid s) \propto \pi(\alpha \mid s) \exp(\tilde{Q}^\pi(s, a, g) / \eta) \approx \pi(\alpha \mid s, g) \) for the M-step. In fact, when \( \eta \) is large and the update to \( q(a \mid s) \) from \( \pi(\alpha \mid s, g) \) becomes infinitesimal, the E-step is equivalent to policy gradients \([26, 24]\), which suffers from the sparsity of rewards as discussed in Section 2.

**Hindsight variational distribution.** Maximizing the first term of the ELBO is challenging when rewards are sparse. This motivates choosing a \( q(\tau, g) \) which puts more weights on maximizing the first term. Now, we formally introduce hindsight variational distribution \( q_h(\tau, g) \), the sampling distribution employed equivalently in HER \([2]\). Sampling from this distribution is implicitly defined by an algorithmic procedure:

**Step 1.** Collect an on-policy trajectory or sample a trajectory from a replay buffer \( \tau \sim \mathcal{D} \).

**Step 2.** Find the \( g \) such that the trajectory is rewarding, in that \( R(\tau, g) \) is high or the trial is successful. Return the pair \( (\tau, g) \).

Note that Step 2 can be conveniently carried out with access to the reward function \( r(s, a, g) \) as in \([2]\). Contrary to \( q_{\text{ent}}(\tau, g) \), this hindsight variational distribution maximizes the first term in Equation (11) by construction. This naturally satisfies \( (P.2) \) as \( q_h(\tau, g) \) provides highly rewarding samples \( (\tau, g) \) and hence dense signals to the M-step. The following theorem shows how \( q_h(\tau, g) \) improves the sampling performance of our gradient estimates

**Theorem 2.** \( (\text{Proof in Appendix B.3}) \) Consider the illustrative example in Theorem 1. Let \( h^{\|}(a, g) = r(s, b, g') \nabla_{L_{a,g}} \log \pi(b \mid s, g') / k \) be the normalized one-sample REINFORCE gradient estimator where \( (b, g') \) are sampled from the hindsight variational distribution with an on-policy buffer. Then the relative error \( \sqrt{\text{MSE}[h^{\|}_{a,g}] / \mathbb{E}[h_{a,g}]} = \sqrt{k(1 + o(1))} \) grows sub-linearly for all \( \forall a \in \mathcal{A}, g \in \mathcal{G} \).

Theorem 2 implies that to further reduce the relative error of the hindsight estimator \( h^{\|}(a, g) \) with traditional Monte Carlo sampling would require \( m \approx (\sqrt{k})^2 = k \) samples, which scales linearly with the problem size \( k \). This is a sharp contrast to \( m \approx k^2 \) from using the on-policy REINFORCE gradient
estimator. The above result shows the benefits of IS, where under \( q_h(t, g) \) rewarding trajectory-goal pairs are given high probabilities and this naturally alleviates the issue with sparse rewards. The following result shows that \( q_h(t, g) \) also satisfies (P1) under mild conditions.

**Theorem 3.** (Proof in Appendix B.4.) Assume \( p(g) \) to be uniform without loss of generality and a tabular representation of policy \( \pi_\theta \). At iteration \( t \), assume that the partial E-step returns \( q_h(t, g) \) and the M-step objective in Equation (10) is optimized fully. Also assume the variational distribution to be the hindsight variational distribution \( q_h(t, g) \). Let \( \tilde{p}_t(g) := \int q_h(t, g) \, dt \) be the marginal distribution of goals. The performance is lower bounded as \( J(\pi_{\theta_{n+1}}) \geq |\text{supp}(\tilde{p}_t(g))|/|G| = \tilde{L}_t \). When the replay buffer size \( D \) increases over iterations, the lower bound improves \( \tilde{L}_{t+1} \geq \tilde{L}_t \).

3.1 Algorithm

We now present \( hEM \), a combination of the above E- and M-steps. The algorithm maintains a policy \( \pi_\theta(a \mid s, g) \). At each iteration, \( hEM \) collects \( N \) trajectory-goal pairs by first sampling a goal \( g \sim p(\cdot) \) and then rolling out a trajectory \( \tau \). All trajectories are stored into a replay buffer \( \mathcal{D} \) [3]. At training time, \( hEM \) carries out a partial E-step by sampling \((\tau, g)\) pairs from \( q_h(\tau, g) \). For the partial M-step, the policy is updated through several gradient ascents on Equation (10) with the Adam optimizer [30]. Importantly, \( hEM \) is an off-policy RL algorithm without value functions, which also makes it agnostic to reward functions. The pseudocode is summarized in Algorithm 1. Please refer to Appendix C for full descriptions of the algorithm.

**Algorithm 1** Hindsight Expectation Maximization (hEM)

1: **INPUT** policy \( \pi_\theta(a \mid s, g) \).
2: while \( t = 0, 1, 2... \) do
3:   Sample goal \( g \sim p(\cdot) \) and trajectory \( \tau \sim p(\cdot \mid \theta, g) \) by executing \( \pi_\theta \) in the MDP. Save data \((\tau, g)\) to a replay buffer \( \mathcal{D} \).
4:   **E-step.** Sample from \( q_h(\tau, g) \): sample \( \tau \equiv (s_t, a_t)_{t=0}^{T-1} \sim \mathcal{D} \) and find rewarding goals \( g \).
5:   **M-step.** Update the policy by a few gradient ascents \( \theta \leftarrow \theta + \nabla_{\theta} \log \sum_{t=0}^{T-1} \log p_\theta(a_t \mid s_t, g) \).
6: end while

3.2 Connections to prior work

**Hindsight experience replay.** The core of HER lies in the hindsight goal replay [2]. Similar to \( hEM \), HER samples trajectory-goal pairs from the hindsight variational distribution \( q_h(\tau, g) \) and minimize the Q-learning loss \( \mathbb{E}_{(\tau, g) \sim q_h(\cdot)} \left[ \sum_{t=0}^{T-1} (Q_\theta(s_t, a_t, g) - r(s_t, a_t, g) - \gamma \max_{a'} Q_\theta(s_t, a', g))^2 \right] \). The development of \( hEM \) in Section 3 formalizes this choice of the sampling distribution \( q(\tau, g) := q_h(\tau, g) \) as partially maximizing the ELBO during an E-step. Compared to \( hEM \), HER learns a critic \( Q_\theta(s, a, g) \). We will see in the experiments that such critic learning tends to be much more unstable when rewards are sparse and inputs are high-dimensional, as was also observed in [8, 9].

**Hindsight policy gradient.** In its vanilla form, the hindsight policy gradient (HPG) considers on-policy stochastic gradient estimators of the RL objective [24] as \( \mathbb{E}_{p(g)p(\tau \mid \theta, g)}[R(\tau, g) \nabla_{\theta} \log p(\tau \mid \theta, g)] \). Despite variance reduction methods such as control variates [26, 24], the unbiased estimators of HPG do not address the rare event issue central to sparse rewards MDP, where \( R(\tau, g) \nabla_{\theta} \log p(\tau \mid \theta, g) \) taking non-zero values is a rare event under the on-policy measure \( p(g)p(\tau \mid \theta, g) \). Contrast HPG to the unbiased IS objective in Equation (8): \( \mathbb{E}_{q(\tau, g)}[R(\tau, g) \nabla_{\theta} \log p(\tau \mid \theta, g)] \cdot \frac{p_\theta(\tau \mid \theta, g)}{q(\tau, g)} \), where the proposal \( q(\tau, g) \) ideally prioritizes the rare events [5] to generate rich learning signals. \( hEM \) further avoids the explicit IS ratios with the variational approach that leads to an ELBO [7].

4 Experiments

We evaluate the empirical performance of \( hEM \) on a wide range of goal-conditioned RL benchmark tasks. These tasks all have extremely sparse binary rewards which indicate success of the trial. The evaluation criterion is the success rate at test time. Since \( hEM \) builds on pure model-free concepts, we focus on the model-free state-of-the-art algorithm HER [2] as a comparison. In some cases we
also compare with closely related HPG [24]: however, we find that even as HPG adopts more dense rewards, its performance evaluated as the success rate is much more inferior than HER and iEM. For hyper-parameter details and additional results, please see Appendix C.

4.1 Simple examples

Flip bit. Taken from [2], the MDP is parameterized by the number of bits $K$. The state space and goal space $S = G = \{0, 1\}^K$ and the action space $A = \{1, 2, \ldots, K\}$. Given $s_t$, the action flips the bit at location $a_t$. The reward function is $r(s_t, a_t) = \mathbb{I}[s_{t+1} = g]$, the state is flipped to match the target bit string. The environment is difficult for traditional RL methods as the search space is of exponential size $|S| = 2^K$. In Figure 3(a), we present results for HER (taken from Figure 1 of [2]), iEM and HPG. Observe that iEM and HER consistently perform well even when $K = 50$ while the performance of HPG drops drastically as the underlying spaces become enormous. See Figure 3(b) for the training curves of iEM and HPG for $K = 50$; note that HPG does not make any progress.

Continuous navigation. As a continuous analogue of the Flip bit MDP, consider a $K$-dimensional navigation task with a point mass. The state space and goal space coincide with $X = \mathcal{G} = [-1,1]^K$ while the actions $A = [-0.2, 0.2]^K$ specify changes in states. The reward function is $r(s_t, a_t) = \mathbb{I}[\|s_{t+1} - g\| < 0.1]$ which indicates success when reaching the goal location. Results are shown in Figure 3(c) where we see that as $K$ increases, the search space quickly explodes and the performance of HER degrades drastically. The performance of iEM is not greatly influenced by increases in $K$. See Figure 3(d) for the comparison of training curves between iEM and HER for $K = 40$. HER already learns much more slowly compared to iEM and degrades further when $K = 80$.

4.2 Goal-conditioned reaching tasks

To assess the performance of iEM in contexts with richer system dynamics, we consider a wide range of goal-conditioned reaching tasks. We present details of their state space $X$, goal space $G$ and action space $A$ in Appendix C. These include Point mass, Reacher goal, Fetch robot and Sawyer robot, as illustrated in Figure 7 in Appendix C.

Across all tasks, the reward takes the sparse form $r(s, a, g) = \mathbb{I}[\text{success}]$. As a comparison, we also include a HER baseline where the rewards take the form $\hat{r}(s, a, g) = -\mathbb{I}[\text{failure}]$. Such reward shaping does not change the optimality of policies as $\hat{r} = r - 1$ but makes the reward more ‘dense’ and is more suitable for learning by neural network based Q-functions. We observe that this makes a significant difference in the performance of HER. We denote this HER baseline as ‘HER-dense’.

From the result in Figure 4, we see that iEM performs significantly better than HER with binary rewards (HER-sparse). The performance of iEM quickly converges to optimality while HER struggles at learning good Q-functions. However, when compared with HER-dense, iEM does not achieve noticeable gains. Such an observation confirms the importance of reward shaping to HER.

4.3 Image-based tasks

We further assess the performance of iEM when policy inputs are high-dimensional images (see Figure 8 for illustrations). Across all tasks, the state inputs are by default images $s \in \mathbb{R}^{w \times h \times 3}$ where $w \in \{48, 84\}$ while the goal is still low-dimensional. See Appendix C for the network architectures.
We focus on the comparison between \( hEM \) and HER-dense in Figure 5, as the performance of HER with binary rewards is inferior as seen Section 4.2. We see that for image-based tasks, HER-dense significantly underperforms \( hEM \). While HER-dense makes slow progress for most cases, \( hEM \) achieves stable learning across all tasks. We speculate this is partly due to the common observations [8, 9] that TD-learning directly from high-dimensional image inputs is challenging. For example, prior work [31] has applied a variational autoencoder approach [12] to reduce the dimension of the image inputs for downstream TD-learning. On the contrary, \( hEM \) only requires optimization in a supervised learning style, which is much more stable with end-to-end training on image inputs.

Further, image-based goals are much easier to specify in certain contexts [31]. We evaluate \( hEM \) on image-based goals for the Sawyer robot and achieve similar performance as the state-based goals. See results in Figure 9 in Appendix C.

4.4 Ablation study

\( hEM \) collects \( N \) trajectories at each iteration, which is set to \( N = 20 \) in previous experiments (except for Fetch robot (I) and Reacher (I) where \( N = 80 \)). In certain cases, we find that the performance of \( hEM \) critically depends on the size of \( N \). In Figure 9 in Appendix C we provide ablation results on the Flip bit MDP with \( K = 50 \) and image-based Fetch robot task. Both tasks are challenging, both due to an enormous state space or the high dimensionality of the images. In general, we find that larger \( N \) leads to better performance. Similar observations have been made for HER [2], where increasing the number of parallel workers generally improves training performance.

5 Conclusion

We present a probabilistic framework for goal-conditioned RL. This framework motivates the development of \( hEM \), a simple and effective off-policy RL algorithm. Our formulation draws formal connections between hindsight goal replay [2] and IS for rare event simulation. \( hEM \) combines the stability of supervised learning updates via the M-step and the hindsight replay technique via the E-step. We show improved performance over a variety of benchmark RL tasks, especially in high-dimensional input settings with sparse binary rewards.
References

[1] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In ICML, volume 99, pages 278–287, 1999.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McCreight, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In Advances in neural information processing systems, pages 5048–5058, 2017.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

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

[5] Gerardo Rubino and Bruno Tuffin. Rare event simulation using Monte Carlo methods. John Wiley & Sons, 2009.

[6] George Casella and Roger L Berger. Statistical inference, volume 2. Duxbury Pacific Grove, CA, 2002.

[7] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. Journal of the American Statistical Association, 112(518):859–877, 2017.

[8] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. arXiv preprint arXiv:1907.00953, 2019.

[9] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. arXiv preprint arXiv:2004.13649, 2020.

[10] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, volume 1. MIT press Cambridge, 1998.

[11] Todd K Moon. The expectation-maximization algorithm. IEEE Signal processing magazine, 13(6):47–60, 1996.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[13] Rajesh Ranganath, Sean Gerrish, and David M Blei. Black box variational inference. In Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, 2014.

[14] Rudolf Kalman. On the general theory of control systems. IRE Transactions on Automatic Control, 4(3):110–110, 1959.

[15] Emanuel Todorov. General duality between optimal control and estimation. In Decision and Control, 2008. CDC 2008. 47th IEEE Conference on, pages 4286–4292. IEEE, 2008.

[16] Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In Advances in neural information processing systems, pages 849–856, 2009.

[17] Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In Advances in Neural Information Processing Systems, pages 207–215, 2013.

[18] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. arXiv preprint arXiv:1806.06920, 2018.
[19] H Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruba Tirumala, et al. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. arXiv preprint arXiv:1909.12238, 2019.

[20] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

[21] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. arXiv preprint arXiv:1804.02808, 2018.

[22] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. arXiv preprint arXiv:1805.00909, 2018.

[23] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010.

[24] Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Juergen Schmidhuber. Hindsight policy gradients. arXiv preprint arXiv:1711.06006, 2017.

[25] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In Reinforcement Learning, pages 5–32. Springer, 1992.

[26] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems, pages 1057–1063, 2000.

[27] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. Nature, 550(7676):354–359, 2017.

[28] Quan Vuong, Yiming Zhang, and Keith W Ross. Supervised policy update for deep reinforcement learning. arXiv preprint arXiv:1805.11706, 2018.

[29] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. arXiv preprint arXiv:1911.08265, 2019.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[31] Ashvin V 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, pages 9191–9200, 2018.

[32] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290, 2018.

[33] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In Aaai, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

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

[35] Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 243–252. JMLR. org, 2017.

[36] Michael C Fu. Stochastic gradient estimation. In Handbook of simulation optimization, pages 105–147. Springer, 2015.
[37] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[38] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[39] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. In *Advances in Neural Information Processing Systems*, pages 15298–15309, 2019.

[40] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, pages 14814–14825, 2019.

[41] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[42] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. *https://github.com/openai/baselines*, 2017.
A Details on Graphical Models for Reinforcement Learning

In this section, we review details of the RL as inference framework [32, 22] and highlight its critical differences from Variational RL.

The graphical model for RL as inference is shown in Figure 6(c). The framework also assumes a trajectory variable \( \tau \equiv (s_t, a_t)_{t=0}^{T-1} \) which encompasses the state and action sequences. Conditional on the trajectory variable \( \tau \), the optimality variable is defined as \( p(O = 1 \mid \tau) \propto \exp(\sum_{t=0}^{T-1} r(s_t, a_t) / \alpha) \) for \( \alpha > 0 \). Under this framework, the trajectory variable has a prior \( a_t \sim p(\cdot) \) where \( p(\cdot) \) is usually set to be a uniform distribution over the action space \( \mathcal{A} \).

The policy parameter \( \theta \) comes into play with the inference model. The framework asks the question: what is the posterior distribution \( p(\tau \mid O = 1) \). To approximate this intractable posterior distribution, consider the variational distribution \( q(\tau) := \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t)p(s_{t+1} \mid s_t, a_t) \). Searching for the best approximate posterior by minimizing the KL-divergence \( \text{KL}(q(\tau) \parallel p(\tau \mid O = 1)) \), it can be shown that this is equivalent to maximum-entropy RL [33–35]. It is important to note that RL as inference does not contain trainable parameters for the generative model.

Contrasting this to Variational RL and the graphical model for goal-conditioned RL in Figure 2: the policy dependent parameter \( \theta \) is part of a generative model. The variational distribution \( q(\tau, g) \), defined separately from \( \theta \), is the inference model. In such cases, the variational distribution \( q(\tau, g) \) is an auxiliary distribution which aids in the optimization of \( \theta \) by performing partial E-steps.

![Figure 6: Plot (c) shows the graphical model for RL as inference [32, 22]. Solid lines represent generative models and dashed lines represent inference models. Circles represent random variables and squares represent parameters. Filled circles represent observed random variables. This graphical model does not have trainable parameters for the generative model. The policy dependent parameter \( \theta \) is in the inference model.](image)

B Details on proof

B.1 Proof of Proposition Proposition 1

The proof follows from the observation that \( p(O = 1) = \mathbb{E}_{g \sim p(\cdot)} p(\tau = 1 \mid \tau, g) = J(\pi_\theta) \), and taking the log does not change the optimal solution.

B.2 Proof of Theorem 1

Recall that we have a one-step MDP setup where \( \mathcal{A} = \mathcal{G} \) and \( |\mathcal{A}| = k \). The policy \( \pi(a \mid s, g) = \text{softmax}(L_{a,g}) \) is parameterized by logits \( L_{a,g} \). When the policy is initialized randomly, we have \( L_{a,g} \equiv L \) for some \( L \) and \( \pi(a \mid s, g) = 1/k \) for all \( a, g \). Assume also \( p(g) = 1/k, \forall g \).

The one-sample REINFORCE gradient estimator for the component \( L_{a,g} \) is \( \eta_{a,g} = r(s, b, g') \log L_{a,g} \pi(b \mid s, g') \). With \( g' \sim p(\cdot) \) and \( b \sim \pi(\cdot \mid s, g') \). Further, we can show

\[
\mathbb{E}[\eta_{a,g}] = \frac{1}{k^2} \delta_{a,g} - \frac{1}{k^3}, \quad \mathbb{V}[\eta_{a,g}] = \left( \frac{1}{k^2} + \frac{2}{k^5} - \frac{2}{k^3} - \frac{1}{k^4} \right) \delta_{a,g} + \frac{1}{k^4} - \frac{1}{k^6},
\]

where \( \delta_{a,b} \) are dirac-delta functions, which mean \( \delta_{a,b} = 1 \) if \( a = b \) and \( \delta_{a,b} = 0 \) otherwise. Taking the ratio, we have the squared relative error (note that the estimator is unbiased and MSE consists purely of the variance)

\[
\frac{\text{MSE}[\eta_{a,g}]}{\mathbb{E}[\eta_{a,g}]^2} = \frac{(k^4 + o(k^4))\delta_{a,g} + (k^2 + o(k^2))}{(k^2 + o(k^2))\delta_{a,g} + 1}
\]
The expression takes different forms based on the delta-function $\delta_{a,g}$. However, in either case (either $\delta_{a,g} = 1$ or $\delta_{a,g} = 0$), it is clear that $\frac{\text{MSE}[\eta_{a,g}]}{\mathbb{E}[\eta_{a,g}]} = k^2(1 + o(1))$, which directly reduces to the result of the theorem.

**Comment on the control variates.** We also briefly study the effect of control variates. Let $X, Y$ be two random variables and assume $\mathbb{E}[Y] = 0$. Then compare the variance of $\mathbb{V}[X]$ and $\mathbb{V}[X + \alpha Y]$ where $\alpha$ is chosen optimally to minimize the variance of the second estimator. It can be shown that with the best $\alpha^*$, the ratio of variance reduction is $(\mathbb{V}[X] / \mathbb{V}[X + \alpha^* Y]) / \mathbb{V}[X] = \rho^2 := \text{Cov}^2[X, Y] / \mathbb{V}[X] \mathbb{V}[Y]$. Consider the state-based control variate for the REINFORCE gradient estimator, in this case $-\alpha \cdot \nabla_{L_{a,g}} \log \pi(b \mid s, g' \mid s, g')$ where $\alpha$ is chosen to minimize the variance of the following aggregate estimator

$$\eta_{a,g}(\alpha) = r(s, b, g') \log \pi(b \mid s, g') - \alpha \log L_{a,g} \pi(b \mid s, g').$$

Note that in practice, $\alpha$ is chosen to be state-dependent for REINFORCE gradient estimator of general MDPs and is set to be the value function $\alpha := V^\pi(s)$. Such a choice is not optimal [36] but is conveniently adopted in practice. Here, we consider an optimal $\alpha^*$ for the one-step MDP. The central quantity is the squared correlation $\rho^2$ between $r(s, b, g') \log \pi(b \mid s, g')$ and $\log L_{a,g} \pi(b \mid s, g')$. With similar computations as above, it can be shown that $\rho^2 \approx 1$ for $b \neq g'$ and $\rho^2 \approx \frac{1}{\mathbb{I}}$ otherwise. This implies that for $k$ out of $k^2$ logits parameters, the variance reduction is significant; yet for the rest of the $k^2 - k$ parameters, the variance reduction is negligible. Overall, the analysis reflects that conventional control variates do not address the issue of sup-linear growth of relative errors as a result of sparse gradients.

**B.3 Proof of Theorem 2**

Similar to the proof of Theorem 1, we can show that the normalized one-step REINFORCE gradient estimator $\eta_{a,g} = r(s, b, g') \nabla_{L_{a,g}} \log \pi(b \mid s, g') / k$ with $(b, g') \sim q_k(\tau, g)$ has the following property

$$\frac{\text{MSE}[\eta_{a,g}]}{\mathbb{E}[\eta_{a,g}]} = \frac{(k^3 + o(k^3)) \delta_{a,g} + (k + o(k^2))}{(k^2 + o(k)) \delta_{a,g} + 1}.$$ 

This implies the result of the theorem.

**B.4 Proof of Theorem 3**

Without loss of generality we assume $p(g)$ is a uniform measure, i.e. $p(g) = 1/|G|$. If not, we could always find a transformation $g = f(g')$ such that $g'$ takes a uniform measure [6] and treat $g'$ as the goal to condition on.

Let $|G| < \infty$ and recall $\text{supp}(\hat{p}(g))$ to be the support of $\hat{p}(g)$. The uniform distribution assumption deduces that $|\text{supp}(\hat{p}(g))| = \int_{g \in \text{supp}(\hat{p}(g))} dg$. At iteration $t$, under tabular representation, the M-step update implies that $\pi_\theta$ learns the optimal policy for all $g$ that could be sampled from $q(\tau, g)$, which effectively corresponds to the support of $\hat{p}(g)$. Formally, this implies $\mathbb{E}_{p(\tau | \theta_{t+1}, g)}[R(\tau, g)] = 1$ for $\forall g \in \text{supp}(\hat{p}(g))$. This further implies

$$J(\pi_{\theta_{t+1}}) := \int \mathbb{E}_{p(\tau | \theta_{t+1}, g)}[R(\tau, g)] p(g) dg \geq \int_{g \in \text{supp}(\hat{p}(g))} 1 \cdot p(g) dg = |\text{supp}(\hat{p}(g))|/|G|.$$

**C Additional Experiment Results**

**C.1 Details on Benchmark tasks**

All reaching tasks are built with physics simulation engine MuJoCo [37]. We build customized point mass environment; the Reacher and Fetch environment is partly based on OpenAI gym environment [38]; the Sawyer environment is based on the multiworld open source code [https://github.com/vitchyr/multiworld](https://github.com/vitchyr/multiworld). All simulation tasks below have a maximum episode length of $T = 50$. The episode terminates early if the goal is achieved at a certain step. The sparse binary reward function is $r(s, \alpha, g) = \mathbb{I}([\text{success}], $
which indicates the success of the transitioned state $s' = f(s, a)$\(^1\). Below we describe in details the setup of each task, in particular the success criterion.

- **Point mass** [39]. The objective is to navigate a point mass through a 2-D room with obstacles to the target location. $|S| = 4$, $|G| = 2$ and $|A| = 2$. The goals $g \in \mathbb{R}^2$ are specified as a 2-D point on the plane. Included in the state $s$ are the 2-D coordinates of the point mass, denoted as $s_{xy} \in \mathbb{R}^2$. The success is defined as $d(z(s_{xy}), z(g)) \leq d_0$ where $d(\cdot, \cdot)$ is the Euclidean distance, $z(g)$ is an element-wise normalization function $z(x) := (x - x_{\text{min}})/(x_{\text{max}} - x_{\text{min}})$ where $x_{\text{max}}, x_{\text{min}}$ are the boundaries of the wall. The normalized threshold is $d_0 = 0.02 \cdot \sqrt{2}$.

- **Reacher** [38]. The objective is to move via joint motors the finger tip of a 2-D Reacher robot to reach a target goal location. $|S| = 11$, $|G| = 2$ and $|A| = 2$. As with the above point mass environment, the goals $g \in \mathbb{R}^2$ are locations of a point at the 2-D plane. Included in the state $s$ are 2-D coordinates of the finger tip location of the Reacher robot $s_{xy}$. The success criterion is defined identically as the point mass environment.

- **Fetch robot** [38, 2]. The objective is to move via position controls the end effector of a fetch robot, to reach a target location in the 3-D space. $|S| = 10$, $|G| = 3$ and $|A| = 3$. This task belongs to the standard environment in OpenAI gym [38] and we leave the details to the code base and [2].

- **Sawyer robot** [31, 40]. The objective is to move via motor controls of the end effector of a sawyer robot, to reach a target location in the 3-D space. $|S| = |G| = |A| = 3$. This task belongs to the multiworld code base.

**Details on image inputs.** For the customized point mass and Reacher environments, the image inputs are taken by cameras which look vertically down at the systems. For the Fetch robot and Sawyer robot environment, the images are taken by cameras mounted to the robotic systems. See Figure 8 for an illustration of the image inputs.

---

\(^1\)For such simulation environments, the transition $s' \sim p(\cdot \mid s, a)$ is deterministic so we equivalently write $s' = f(s, a)$ for some deterministic function $f$. 
C.2 Details on Algorithms and Hyper-parameters

**Hindsight Expectation Maximization.** In our implementation, we take the policy network $\pi_\theta(a \mid s, g)$ to be a state-goal conditional Gaussian distribution $\pi_\theta(a \mid s, g) = \mathcal{N}(\mu_\theta(s, g), \sigma^2)$ with a parameterized mean $\mu_\theta(s, g)$ and a global standard deviation $\sigma^2$. The mean is the takes the concatenated vector $[x, g]$ as inputs, has $3 \times 5$ hidden layers each with $3 \times 5$ hidden units interleaved with relu($x$) non-linear activation functions, and outputs a vector $\mu_\theta(s, g) \in \mathbb{R}^{|A|}$.

hEM alternates between data collection using the policy and policy optimization with the EM-algorithms. During data collection, the output action is perturbed by a Gaussian noise $a' = \mathcal{N}(0, \sigma_n^2) + a, \ a \sim \pi_\theta(\cdot \mid s, g)$ where the scale is $\sigma_n = 0.5$. Note that injecting noise to actions is a common practice in off-policy RL algorithms to ensure sufficient exploration [3, 4]. The baseline hEM collects data with $N = 30$ parallel MPI actors, each with $k = 20$ trajectories. When sampling the hindsight goal given trajectories, we adopt the future strategy specified in HER [2]: in particular, at state $s$, future achieved goals are uniformly sampled at trainig time as $q_h(\tau, g)$. All parameters are optimized with Adam optimizer [30] with learning rate $\alpha = 10^{-3}$. By default, we run $M = 30$ parallel MPI workers for data collection and training, at each iteration hEM collects $N = 20$ trajectories from the environment. For image-based reacher and Fetch robot, hEM collects $N = 80$ trajectories.

**Hindsight Experience Replay.** By design in [2], HER is combined with off-policy learning algorithms such as DQN or DDPG [3, 4]. We describe the details of DDPG. The algorithm maintains a Q-function $Q_\theta(s, a, g)$ parameterized similarly as a universal value function [41]: the network takes as inputs the concatenated vector $[x, a, g]$, has $3 \times 5$ hidden layers with $h = 256$ hidden units per layer interleaved with relu($x$) non-linear activation functions, and outputs a single scalar. The policy network $\pi_\theta(s, g)$ takes the concatenated vector $[x, g]$ as inputs, has the same intermediate architecture as the Q-function network and outputs the action vector $\pi_\theta(s, g) \in \mathbb{R}^{|A|}$. We take the implementation from OpenAI baseline [42], all missing hyper-parameters are the default hyper-parameters in the code base. Across all tasks, HER is run with $M = 20$ parallel MPI workers as specified in [42].

**Image-based architecture.** When state or goal are image-based, the Q-function network/policy network applies a convolutional network to extract features. For example, let $s, g \in \mathbb{R}^{w \times h \times 3}$ where $w \in \{48, 84\}$ be raw images, and let $f_\theta(s), f_\theta(g)$ be the features output by the convolutional network. These features are concatenated before passing through the fully-connected networks described above. The convolutional network has the following architecture: $[32, 8, 4] \rightarrow \text{relu} \rightarrow [64, 4, 2] \rightarrow \text{relu} \rightarrow [64, 3, 2] \rightarrow \text{relu}$, where $[n_f, r_f, s_f]$ refers to: $n_f$ number of feature maps, $r_f$ feature patch dimension and $s_f$ the stride.

C.3 Ablation study

**Ablation study on the effect of $N$.** hEM collects $N$ trajectories at each training iteration. We vary $N \in \{5, 10, 20, 40, 80\}$ on two challenging domains: Flip bit $K = 50$ and Fetch robot (image-based) and evaluate the corresponding performance. See Figure 9. We see that in general, large $N$ tends to lead to better performance. For example, when $N = 5$, hEM learns slowly on Flip bit; when $N = 80$, hEM generally achieves faster convergence and better asymptotic performance across both tasks. We speculate that this is partly because with large $N$ the algorithm can have a larger coverage over goals (larger support over goals in the language of Theorem 3). With small $N$, the policy might converge prematurely and hence learn slowly. Similar observations have been made for HER, where they find that the algorithm performs better with a large number of MPI workers (effectively large $N$).

**Ablation on image-based goals.** To further assess the robustness of hEM against image-based inputs, we consider Sawyer robot where both states and goals are image-based. This differs from experiments shown in Figure 5 where only states are image-based. In Figure 9(c), we see that the performance of hEM does not degrade even when goals are image-based and is roughly agnostic to the size of the image. Contrast this with HER, which does not make significant progress even when only states are image-based.
**C.4 Comparison between $hEM$ and HPG**

We do not list HPG as a major baseline for comparison in the main paper, primarily due to a few reasons: by design, the HPG agent tackles discrete action space (see the author code base [https://github.com/paulorauber/hpg](https://github.com/paulorauber/hpg)), while many goal-conditioned baselines of interest [2, 31, 40] are continuous action space. Also, in [24] the author did not report comparison to traditional baselines such as HER and only report cumulative rewards instead of success rate as evaluation criterion. Here, we compare $hEM$ with HPG on a few discrete benchmarks provided in [24] to assess their performance.

**Details on HPG.** The HPG is based on the author code base. [24] proposes several HPG variants with different policy gradient variance reduction techniques [26] and we take the HPG variant with the highest performance as reported in the paper. Throughout the experiment we set the learning rate to be $10^{-3}$ and other hyper-parameters take default values.

**Benchmarks.** We compare $hEM$ and HPG on Flip bit $K = 25$, 50 and the four room environment. The details of the Flip bit environment could be found in the main paper. The four room environment is used as a benchmark task in [24], where the agent navigates a grid world with four rooms to reach a target location within episodic time limit. The agent has access to four actions, which moves the agent in four directions. The trial is successful only if the agent reaches the goal in time.

**Results.** We show results in Figure 10. For the Flip bit $K = 25$, HPG and $hEM$ behave similarly: both algorithms reach the near-optimal performance quickly and has similar convergence speed; when the state space increases to $K = 50$, HPG does not make any progress while the performance of $hEM$ steadily improves. Finally, for the four room environment, we see that though the performance of HPG initially increases quickly as $hEM$, its success rate quickly saturates to a level significantly below the asymptotic performance of $hEM$. These observations show that $hEM$ performs much more robustly and significantly better than HPG, especially in challenging environments.

![Figure 9](image)

**Figure 9:** Ablation study. Plot (a) and (b): The effect of the data collection size $N$. Plot (c): The effect of image-based inputs for both states and goals. ‘$hEM$-48’ refers to image-based inputs with size $48 \times 48 \times 3$.

![Figure 10](image)

**Figure 10:** Comparison between $hEM$ and HPG. HPG performs well on Flip bit MDP with $K = 25$, but when $K = 50$ its performance drops drastically. HPG also underperforms $hEM$ on the four room environment where it makes fast progress initially but saturates to a low sub-optimal level.