

A Theoretical Analysis

A.1 Proof of Theorem 1

Proof. Following GCSL [12], we denote a real trajectory as $\tau = \{s_0, a_0, \dots, s_T, a_T\}$. The expected goal is g , and the real dynamics is $p(s_{t+1}|s_t, a_t)$. We assume the initial state s_0 is uniformly distributed in the state space. To simplify the analysis, the relabeling goal is defined as the achieved goal of the last state $\mathcal{G}(\tau) \triangleq \phi(s_T)$, where ϕ is the state-to-goal mapping and \mathcal{G} is the mapping from trajectories to relabeling goals. This simplification is also adopted in [12]. Then we denote a virtual trajectory generated by the trained dynamics model $p_m(s'_{t+1}|s'_t, a'_t)$ as $\tau_m = \{s'_0, a'_0, \dots, s'_n, a'_n\}$, $s'_0 = s_0$, and suppose $T \geq n$. The model-based relabeling goal is defined as $\mathcal{G}(\tau_m) \triangleq \phi(s'_n)$. We utilize a sparse reward function: $r(s_t, a_t, g) = 1[\phi(s_t) = g]$, therefore the reward at each timestep is bounded $r_t \in [0, 1]$. Other variables are defined in Section 3.

In our setting, the original multi-goal RL objective is:

$$J(\pi) = E_{g \sim p(g), a_t \sim \pi, s_{t+1} \sim p(\cdot|s_t, a_t)} \left[\sum_{t=0}^T \gamma^t r(s_t, a_t, g) \right].$$

Similarly, the expected return of the virtual trajectory τ_m is:

$$J_m(\pi) = E_{g \sim p(g), a_t \sim \pi, s_{t+1} \sim p_m(\cdot|s_t, a_t)} \left[\sum_{t=0}^n \gamma^t r(s_t, a_t, g) \right].$$

Following MBPO [15], we first bound model error using ϵ_m . As our model-based approach collects data with current policy π , the policy shift $\epsilon_\pi = 0$. Next, we apply Lemma 2 to bound $J(\pi)$ with $J_m(\pi)$ and absorb constants into C_1 as:

$$J(\pi) \geq J_m(\pi) + C_1 \epsilon_m,$$

where $C_1 = -\frac{2\gamma r_{max}}{(1-\gamma)^2} = -\frac{2\gamma}{(1-\gamma)^2}$ is derived from Lemma 2.

In GCSL, agents optimize the goal-reaching objective, i.e., maximizing the probability of reaching g at the last step:

$$\hat{J}(\pi) = E_{g \sim p(g), a_t \sim \pi, s_{t+1} \sim p(\cdot|s_t, a_t)} [1[\mathcal{G}(\tau) = g]].$$

With Lemma 3, we can also obtain the goal-reaching return bound for model-based return $J_m(\pi)$:

$$J_m(\pi) \geq \gamma^n E_{g \sim p(g), a_t \sim \pi, s_{t+1} \sim p_m(\cdot|s_t, a_t)} [1[\mathcal{G}(\tau_m) = g]],$$

and denote the inequality as $J_m(\pi) \geq \gamma^n \hat{J}_m(\pi)$. For $\hat{J}_m(\pi)$, we follow Theorem 4.1 in [12] and have:

$$\hat{J}_m(\pi) \geq E_{a_t \sim \pi, s_{t+1} \sim p_m(\cdot|s_t, a_t)} \left[\sum_{t=0}^n \log \pi(a_t | s_t, \mathcal{G}(\tau_m)) \right] + C_2$$

As we collect data using the current policy π , we don't have the policy variation term. The right side of this inequality means using relabeled data for maximum likelihood estimation, therefore the policy π is required to be stochastic and select actions with non-zero probability.

Combining the intermediate results, we can conclude that:

$$\begin{aligned} J(\pi) &\geq J_m(\pi) + C_1 \epsilon_m \geq \gamma^n \hat{J}_m(\pi) + C_1 \epsilon_m \\ &\geq \gamma^n E_{a_t \sim \pi, s_{t+1} \sim p_m(\cdot|s_t, a_t)} \left[\sum_{t=0}^n \log \pi(a_t | s_t, \mathcal{G}(\tau_m)) \right] + C_1 \epsilon_m + C_2 \end{aligned}$$

Then, we take the Diagonal Gaussian policy with mean vector $\pi(s_t, g)$ of dimension $|a|$ and non-zero positive constant variance σ^2 :

$$P(a_t | s_t, g) = \frac{1}{(\sigma \sqrt{2\pi})^{|a|}} e^{-\frac{\|a_t - \pi(s_t, g)\|_2^2}{2\sigma^2}}.$$

Note that we use P_i to represent 'pi' in the Gaussian distribution and to distinguish from the policy π . P_i is a constant and we will absorb it in C_2 in the following derivation. Denoting the model-based relabeled data distribution as B_m and relabeled goal as g' , we have:

$$\begin{aligned} J(\pi) &\geq -\frac{\gamma^n}{2\sigma^2} E_{(a_t, s_t, g')_{t=0}^n \sim B_m} \left[\sum_{t=0}^n \|a_t - \pi(s_t, g')\|_2^2 \right] + C_1 \epsilon_m + C_2 \\ &= -\frac{n\gamma^n}{2\sigma^2} E_{(a_t, s_t, g') \sim B_m} \|a_t - \pi(s_t, g')\|_2^2 + C_1 \epsilon_m + C_2 \end{aligned}$$

This completes the proof. \square

A.2 Useful Lemmas

Lemma 1 (Finite Horizon Return Bound). *Suppose the model error is bounded as $\max_t E_{s \sim p_1^t(s)} D_{TV}(p_1(s'|s, a) \| p_2(s'|s, a)) \leq \epsilon_m$, and $\max_s D_{TV}(\pi_1(a|s) \| \pi_2(a|s)) \leq \epsilon_\pi$. And We denote $J^{(T)}(\pi)$ as expected return of finite horizon T under policy π , and $r_{max} > 0$ is the maximum reward. Then the difference of finite horizon returns are bounded as:*

$$|J^{(T)}(\pi_1) - J^{(T)}(\pi_2)| \leq 2r_{max} \sum_{t=0}^T \gamma^t [t(\epsilon_m + \epsilon_\pi) + \epsilon_\pi]$$

Proof. This lemma can be derived by Lemma B.3 in [15], we also provide a sketch proof.

$$\begin{aligned} |J^{(T)}(\pi_1) - J^{(T)}(\pi_2)| &= \left| \sum_{s,a} \sum_t \gamma^t (p_1^t(s, a) - p_2^t(s, a)) r(s, a) \right| \\ &\leq r_{max} \sum_t \sum_{s,a} \gamma^t |p_1^t(s, a) - p_2^t(s, a)| \\ &= 2r_{max} \sum_t \gamma^t D_{TV}(p_1^t(s, a) \| p_2^t(s, a)) \\ &\leq 2r_{max} \sum_t \gamma^t [D_{TV}(p_1^t(s) \| p_2^t(s)) + \max_s D_{TV}(\pi_1(a|s) \| \pi_2(a|s))] \\ &\leq 2r_{max} \sum_{t=0}^T \gamma^t [t(\epsilon_m + \epsilon_\pi) + \epsilon_\pi] \end{aligned}$$

where $D_{TV}(p_1^t(s) \| p_2^t(s)) \leq t(\epsilon_m + \epsilon_\pi)$ is also prove in Lemma B.3 in [15]. The branched returns bound (Lemma B.3 in [15]) is a special case of our Lemma 1 when horizon $T \rightarrow \infty$. \square

Lemma 2. *Let $J^{(\infty)}(\pi)$ denote $J(\pi)$ with infinite horizon, where the superscript refers to the horizon. $J(\pi) = J^{(T)}(\pi)$, $J_m(\pi) = J_m^{(n)}(\pi)$, $T \geq n$. Given model error ϵ_m and policy shift ϵ_π defined in [15], non-negative reward function $r(s_t, a_t) \geq 0$, and the maximum reward r_{max} , $J(\pi)$ can be bounded as:*

$$J(\pi) \geq J_m(\pi) - \frac{2r_{max}}{(1-\gamma)^2} (\gamma\epsilon_m + 2\epsilon_\pi),$$

Proof. Let π_D denote the data collecting policy, we follow the proof of Theorem A.1 in [15] and have:

$$J^{(T)}(\pi) - J_m^{(T)}(\pi) = \underbrace{J^{(T)}(\pi) - J^{(T)}(\pi_D)}_{L_1} + \underbrace{J^{(T)}(\pi_D) - J_m^{(T)}(\pi)}_{L_2}$$

For L_1 and L_2 , we apply Lemma 1 to derive the following bound (note that there should be no model error in L_1 term):

$$\begin{aligned} J^{(T)}(\pi) - J_m^{(T)}(\pi) &\geq \underbrace{-2r_{max} \sum_{t=0}^T \gamma^t [t\epsilon_\pi + \epsilon_\pi]}_{L_1} - \underbrace{2r_{max} \sum_{t=0}^T \gamma^t [t(\epsilon_m + \epsilon_\pi) + \epsilon_\pi]}_{L_2} \\ &= -2r_{max} \sum_{t=0}^T \gamma^t [t(\epsilon_m + 2\epsilon_\pi) + 2\epsilon_\pi], \end{aligned}$$

To verify the above inequality, let $T \rightarrow \infty$ and leverage the properties $\sum_{t=0}^{\infty} \gamma^t t \leq \frac{\gamma}{(1-\gamma)^2}$ and $\sum_{t=0}^{\infty} \gamma^t \leq \frac{1}{1-\gamma}$, then we can get MBPO performance bound (Theorem A.1 in [15]). Utilizing the assumptions $r_t \in [0, 1]$, $T \geq n$ and properties $\epsilon_m \geq 0$, $\epsilon_\pi \geq 0$, $J_m^{(T)} \geq J_m^{(n)}$, we can then prove the lower bound

$$\begin{aligned} J^{(T)}(\pi) &\geq J_m^{(T)}(\pi) - 2r_{max} \sum_{t=0}^T \gamma^t [t(\epsilon_m + 2\epsilon_\pi) + 2\epsilon_\pi] \\ &\geq J_m^{(T)}(\pi) - 2r_{max} \sum_{t=0}^{\infty} \gamma^t [t(\epsilon_m + 2\epsilon_\pi) + 2\epsilon_\pi] \\ &\geq J_m^{(n)}(\pi) - \frac{2r_{max}[\gamma\epsilon_m + 2\gamma\epsilon_\pi + 2\epsilon_\pi(1-\gamma)]}{(1-\gamma)^2} \\ &\geq J_m^{(n)}(\pi) - \frac{2r_{max}}{(1-\gamma)^2}(\gamma\epsilon_m + 2\epsilon_\pi). \end{aligned}$$

Alternating $J_m^{(n)}(\pi)$, $J^{(T)}(\pi)$ with $J_m(\pi)$, $J(\pi)$ completes the proof. \square

Lemma 3 (Goal-reaching Return Bound). *Given trajectories $\tau = \{s_0, a_0, \dots, s_T, a_T\}$ and reward function $r(s_t, a_t, g) = 1[\phi(s_t) = g]$, the goal-reaching objective $\gamma^T \hat{J}(\pi)$ is a lower bound of the original multi-goal objective $J(\pi)$.*

Proof. Leveraging the definition of $J(\pi)$, $\hat{J}(\pi)$, and the reward function $r(s_t, a_t, g) = 1[\phi(s_t) = g]$, we have:

$$\begin{aligned} J(\pi) &= E_{g \sim p(g), a_t \sim \pi, s_{t+1} \sim p(\cdot | s_t, a_t)} \left[\sum_{t=0}^T \gamma^t r(s_t, a_t, g) \right] \\ &\geq E_{g \sim p(g), a_t \sim \pi, s_{t+1} \sim p(\cdot | s_t, a_t)} \left[\gamma^T r(s_T, a_T, g) \right] \\ &= \gamma^T E_{g \sim p(g), a_t \sim \pi, s_{t+1} \sim p(\cdot | s_t, a_t)} \left[1[\mathcal{G}(\tau) = g] \right] \\ &= \gamma^T \hat{J}(\pi). \end{aligned}$$

\square

A.3 Theoretical Analysis of Virtual Achieved Goals

We provide formal analysis on the expected distance between virtual achieved goals and original desired goals. First, we define the *expected distance* ($ED_\pi(g)$) as the sum of the product of distance and the discounted visitation frequencies:

$$ED_\pi(g) = \sum_s \rho_\pi(s) \|\phi(s) - g\|_2^2$$

where g refer to original desired goals, ϕ is a state-to-goal mapping, and $\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s | \pi) + \gamma^2 P(s_2 = s | \pi) + \dots$, $s_0 \sim p(s_0)$ and actions are sampled according to π . The expected distance $ED_\pi(g)$ measure the proximity of trajectories generated by the policy π and the desired goal g . The following theory demonstrates the expected distance in the model-based relabeling can be minimized by optimizing the policy.

Theorem 2 (Expected Distance Bound, Infinite Horizon). *Suppose the state space is finite, the policy π is optimized to maximize $J(\pi)$, the reward function is sparse $r(s, a, g) = 1[\|\phi(s) - g\| \leq \epsilon]$, ϵ is a small positive threshold, the goal space is bounded, i.e., $\|g - g'\|_2^2 \leq C_g, \forall g, g' \in \mathcal{G}, C_g \geq 0$. Denote the environmental dynamics as $p(s' | s, a)$, the model error ϵ_m , the learned dynamics model as $p_m(s' | s, a)$, and the discounted visitation frequencies with the dynamics model as $\rho_{\pi, m}(s)$. Then, optimizing $J(\pi)$ is equivalent to minimizing the expected distance between virtual achieved goals and original desired goals, i.e., $ED_{\pi, m}(g) = \sum_s \rho_{\pi, m}(s) \|\phi(s) - g\|_2^2$. Specifically, the following inequality holds:*

$$ED_{\pi, m}(g) \leq -C_g \cdot J(\pi) + \frac{C_g + \epsilon}{1 - \gamma} + \frac{2\gamma\epsilon_m C_g}{(1 - \gamma)^2}$$

Proof. We first bound the expected distance as:

$$\begin{aligned}
ED_\pi(g) &= \sum_s \rho_\pi(s) \|\phi(s) - g\|_2^2 \cdot \mathbb{1}[\|\phi(s) - g\|_2^2 \leq \epsilon] + \sum_s \rho_\pi(s) \|\phi(s) - g\|_2^2 \cdot \mathbb{1}[\|\phi(s) - g\|_2^2 > \epsilon] \\
&\leq \frac{\epsilon}{1-\gamma} + \sum_s \rho_\pi(s) \|\phi(s) - g\|_2^2 \cdot \mathbb{1}[\|\phi(s) - g\|_2^2 > \epsilon] \\
&\leq \frac{\epsilon}{1-\gamma} + C_g \sum_s \rho_\pi(s) \cdot \mathbb{1}[\|\phi(s) - g\|_2^2 > \epsilon]
\end{aligned} \tag{8}$$

The above derivation leverages the property that:

$$\sum_s \rho_\pi(s) = \sum_s P(s_0 = s) + \gamma \sum_s P(s_1 = s) + \gamma^2 \sum_s P(s_2 = s) + \dots = \frac{1}{1-\gamma}$$

As in our setting, rewards only depend on states, thus the RL objective can be written as:

$$\begin{aligned}
J(\pi) &= E_{s_0 \sim p(s_0), a_t \sim \pi(a_t | s_t), s_{t+1} \sim p(s_{t+1} | s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \right] \\
&= \sum_{t=0}^{\infty} \sum_s P(s_t = s | \pi) \cdot \gamma^t r(s) = \sum_s \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \pi) \cdot r(s) \\
&= \sum_s \rho_\pi(s) \cdot \mathbb{1}[\|\phi(s) - g\|_2^2 \leq \epsilon] \\
&= \sum_s \rho_\pi(s) \cdot [1 - \mathbb{1}[\|\phi(s) - g\|_2^2 > \epsilon]] \\
&= \frac{1}{1-\gamma} - \sum_s \rho_\pi(s) \cdot \mathbb{1}[\|\phi(s) - g\|_2^2 > \epsilon]
\end{aligned}$$

Taking the upper bound of $ED_\pi(g)$ in Eq 8 into $J(\pi)$, we have:

$$J(\pi) \leq \frac{1}{1-\gamma} - \frac{1}{C_g} [ED_\pi(g) - \frac{\epsilon}{1-\gamma}]$$

and

$$ED_\pi(g) \leq -C_g \cdot J(\pi) + \frac{C_g + \epsilon}{1-\gamma}.$$

The virtual achieved goals are generated with the learned dynamics model, therefore the following inequality also holds for the learned dynamics model:

$$ED_{\pi,m}(g) \leq -C_g \cdot J_m(\pi) + \frac{C_g + \epsilon}{1-\gamma}.$$

From the proof of Lemma 2, $J_m^{(\infty)}(\pi)$ can be bounded as: $J_m(\pi) \geq J(\pi) - \frac{2\gamma\epsilon_m}{(1-\gamma)^2}$, where ϵ_m is the model error defined before. Finally, we can conclude that:

$$\begin{aligned}
ED_{\pi,m}(g) &\leq -C_g \cdot J_m(\pi) + \frac{C_g + \epsilon}{1-\gamma} \\
&\leq -C_g \cdot J(\pi) + \frac{C_g + \epsilon}{1-\gamma} + \frac{2\gamma\epsilon_m C_g}{(1-\gamma)^2}.
\end{aligned}$$

This completes the proof. \square

Theorem 3 (Expected Distance Bound, Finite Horizon). *Assume the finite horizon for model-based rollout equals the horizon in real environment, i.e., $n = T$. Following Theorem 2 and [Ross et al. 2011], the expected distance bound for finite horizon is:*

$$ED_{\pi,m}(g) \leq -C_g \cdot J(\pi) + (C_g + \epsilon)T + 2\gamma\epsilon_m C_g T^2$$

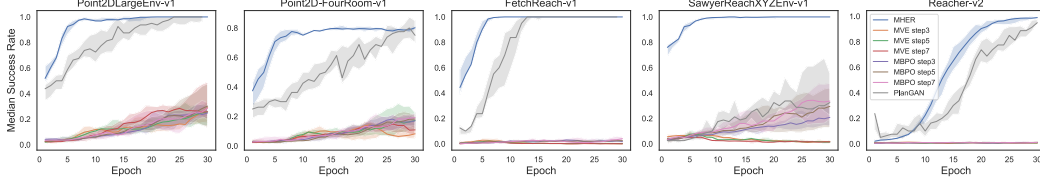


Figure 6: Comparison results with model-based baselines, including PlanGAN.

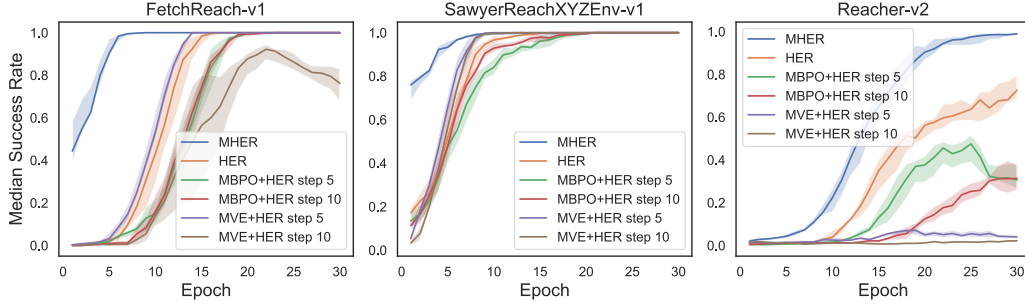


Figure 7: Comparison results with MBPO+HER and MVE+HER in FetchReach-v1, SawyerReachXYZEnv-v1, and Reacher-v2.

B Extra Experimental Results

In this section, we provide additional experimental results to further understand MHER and its effectiveness. We report the average success rate and standard deviation across 10 random seeds for all the experiments.

B.1 Comparison Results with Model-based Baselines

In our paper, we have compared with model-based RL baselines such as MVE [10] and MBPO [15]. Furthermore, we include PlanGAN as our baseline, which is not a RL method but a planning method requiring huge amount of computation. The procedure of selection one single action of planner in PlanGAN is: 1) first random sampling 20 initial actions, 2) further sampling 50 trajectories with model and GANs for each initial action, 3) computing average return of each initial action, and 4) selecting the best action in the initial set. Even if we ignore the additional computation to train GANs, PlanGAN needs at least 10^5 (20 initial action \times 50 trajectories \times 50 steps \times 2 policy and model propagation) times forward propagations than MHER to select a single action. As PlanGAN is highly dependent on the learned model, we train the model and GANs in PlanGAN with a batch size of 128 and each epoch train 25 times. In MHER, we only update the model 2 times each epoch with a batch size of 64, therefore MHER is more computationally friendly compared to PlanGAN. We report the median test success rate and interquartile across 10 random seeds in Figure 6, which apparently shows MHER is more sample efficient than PlanGAN. Moreover, MHER is much more stable compared with PlanGAN and other model-based baselines considering the deviation in the results.

Moreover, we provide comparison results with MBPO+HER and MVE+HER in Figure 7. In the two methods, we perform hindsight relabeling before model-based interaction, and then the model-based interaction is also driven by hindsight goals. Through combining with HER, MBPO and MVE work better in sparse reward tasks. However, their performance still cannot surpasses MHER's.

B.2 Study of the Model Layers

In this section, we study the impact of model layers in MHER by varying the number of layers (each of which is of 256 neurons). The empirical results in Figure 8 show the number of layers does not affect the performance much and even one layer can perform comparably to 4 layers. The dynamics model with 4 layers achieves stable and good performance in different tasks.

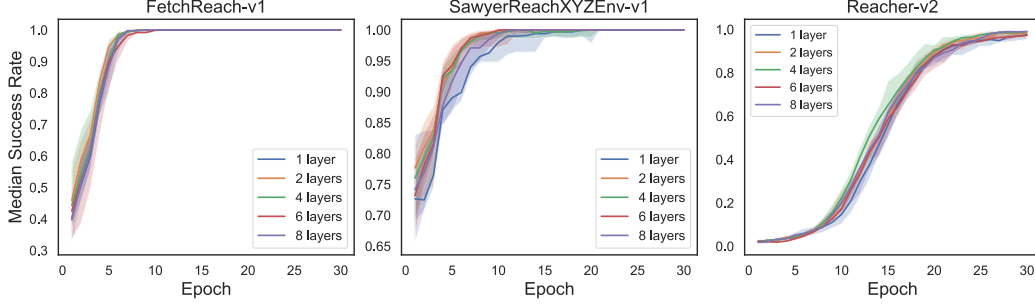


Figure 8: Varying the number of model layers in FetchReach-v1, SawyerReachXYZEnv-v1, and Reacher-v2.

B.3 Parameter Study

Additional parameter studies of α and model-based interaction steps are shown in Figure 9. In the results, we can conclude that the performance of MHER improves when parameter α increases from 0 to 3 in the two environments, but decrease slightly when α is beyond 3 in Reacher-v2. Regarding the model-based interaction steps, we can observe a performance increase as steps increase from 0 to 5, and a performance decrease as steps are more than 5.

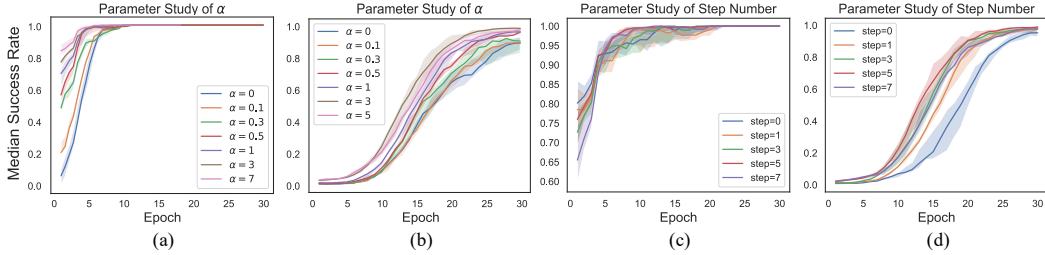


Figure 9: (a)(b) Additional parameter studies of α in SawyerReachXYZEnv-v1 and Reacher-v2. (c)(d) Additional parameter studies of model-based interaction steps in SawyerReachXYZEnv-v1 and Reacher-v2.

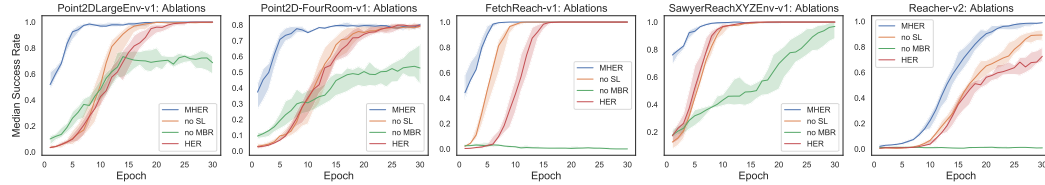


Figure 10: Additional ablation studies in benchmark environments.

B.4 Ablations

We also provide more ablation study results in Figure 10. The results are consistent with the conclusions in our paper: (1) MBR is more important than SL, (2) we can outperform HER using MBR alone, and (3) SL is not robust and fails to learn in FetchReach-v1 and Reacher-v2.

B.5 Additional Comparisons

We conducted additional experiments in Figure 11 to study how does MHER compare to *random relabeling* and HER with *goal noise*. For *random relabeling*, we relabel the transitions with random goals sampled from the goal space. Regarding *goal noise*, Gaussian noise with zero mean and constant (0.01) standard deviation is applied to hindsight goals of HER. We can observe from the results that *goal noise* provides slight improvement over HER in FetchReach-v1 and SawyerReachXYZEnv-v1,

but it worsens the result in Reacher-v2. In addition, it seems that *random relabeling* does not work, which might be because it hardly helps with the sparse reward problem.

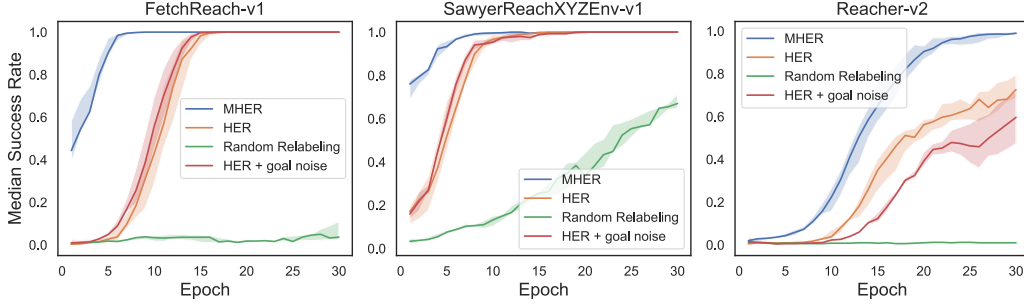


Figure 11: Comparison results between MHER, *random relabeling* and *goal noise*.

C Details about Virtual Achieved Goals

Previous work, HER [1], assumes there exists a state-to-goal mapping function: $\phi : S \rightarrow G$, i.e., given a state s , we can find a goal $g = \phi(s) \in G$ achieved by this state. In the case where each goal corresponds to a state we want to achieve (e.g., 2D point reaching task), goal space equals state space $G = S$ and the mapping ϕ is exactly an identity transformation.

Model-based relabeling is a novel goal relabeling method different from previous relabeling methods and enjoys many advantages by leveraging the dynamics model. MHER follows the same setting as HER. The difference is that MHER leverages the virtual states generated from model-based interaction rather than past collected states. As shown in Figure 12 the blue trajectory is collected by past policy, while the green trajectory is generated by interaction of current policy and the learned dynamics model.

Given a transition in a past collected trajectory $(s_t, a_t, r_t, s_{t+1}, g)$, model-based relabeling (MBR) aims to find a virtual achieved goal to replace the original goal g and reward r_t . MBR interacts with the dynamics model m for n steps starting from s_{t+1} , and collects a virtual trajectory $\{s'_{t+i}, a'_{t+i}, s'_{t+i+1}\}_{i=1}^n$, where $s'_{t+1} = s_{t+1}$, $a'_{t+i} = \pi(s'_{t+i}, g)$, $s'_{t+i+1} = s'_{t+i} + m(s'_{t+i}, a'_{t+i})$, $i \in [1, n]$. Note that we start interaction from s_{t+1} , which is reasonable because starting from s_t means a_t, s_{t+1} also need to be replaced. After model-based interaction, MBR samples from virtual achieved goals $g' = \phi(s_{t+i})$, $i \in [1, n]$ and alternates the original transition as $(s_t, a_t, r'_t, s_{t+1}, g')$, where r'_t is the recomputed reward $r'_t = r(s_t, a_t, g')$ according to Eq. 1. The model-based relabeled data can be further used for off-policy reinforcement learning and goal-conditioned supervised learning.

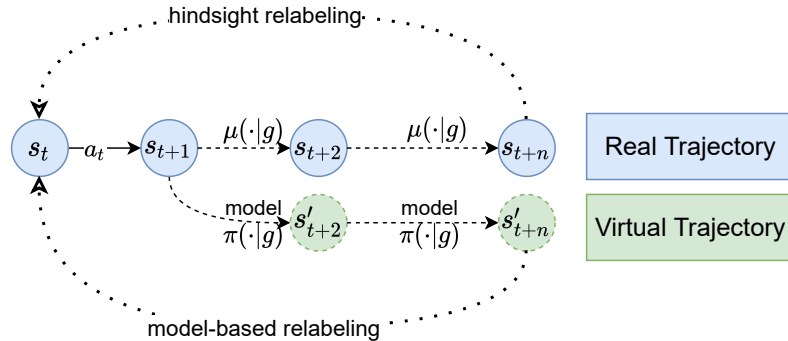


Figure 12: Simplified schematic of model-based relabeling.

D Implementation Details

D.1 Implementation of MHER

In this section, we will provide implementation details of MHER. The actor and critic networks of MHER are both 3-layer fully connected networks with 256 units each and ReLU non-linearities. The actor and critic are updated with Adam optimizer, learning rate 1×10^{-3} . Target nets of actor and critic with the same network structure are adopted for stabilizing training, the Polyak averaging coefficient of the target net is set as 0.9. The probability of random action is 0.3, and the scale of Gaussian noisy for exploration is 0.2. The model-based relabeling rate is 80%, and we keep 20% samples without relabeling. MHER uses one rollout worker, and the buffer size is 10^6 . At the end of each episode (including 100 environment steps), we train the networks with 5 batches of size 64. We only use one single CPU and one GPU to train MHER.

Regarding the dynamics model in MHER, we use a fully connected network with 4 hidden layers and 256 neurons each layer. The optimizer is Adam and the learning rate is 0.001. The dynamics model is trained to minimize the loss \mathcal{L}_{model} in Eq. 4. In the warmup period, we train the dynamics model for 100 updates with a batch size of 512. When training with MHER, every batch we update m with a batch size of 64 for 2 times.

D.2 Implementation of GCSL

We implement GCSL with a Diagonal Gaussian policy with a constant standard deviation of 0.2. GCSL only keeps a single policy network (3 layers, 256 neurons each) as the actor in MHER, without target network. The policy network is trained to minimize the loss

$$\mathcal{L}_{GCSL} = E_{(s_t, a_t, g') \sim B_h} [\|a_t - \pi(s_t, g')\|_2^2]$$

where g' is relabeled using future achieved goals like HER [11]. The optimizer is Adam and the learning rate is also $1 \cdot 10^{-3}$. GCSL shares other parameters with MHER for fair comparison, such as replay buffer size and batch size.

D.3 Implementation of MVE and MBPO

Model-based value expansion (MVE) [10] rollouts with a learned dynamics model and introduces multi-step value estimation based on the expanded transitions. Denote the replay buffer as B , a real transition $(s_t, a_t, r_t, s_{t+1}, g)$, generated transitions $\{\hat{s}_{t+i}, \hat{a}_{t+i}, \hat{r}_{t+i}\}, i \in [1, H], \hat{s}_{t+1} = s_{t+1}$, MVE updates the Q-function to minimize the following loss:

$$\mathcal{L}_{critic} = E_{(s_t, a_t, g, r_t, s_{t+1}) \sim B} \left[\left(\sum_{i=0}^{H-1} \gamma^i \hat{r}_{t+i} + \gamma^H Q(\hat{s}_{t+H}, \pi(\hat{s}_{t+H}, g), g) - Q(s_t, a_t, g) \right)^2 \right], \quad (9)$$

where $\hat{r}_t = r_t, \hat{r}_{t+i} = r(\hat{s}_{t+i}, \pi(\hat{s}_{t+i}, g), g)$, r is the reward function in Eq. 1. We also use DDPG to learn the policy, and the hyper-parameters of DDPG and dynamics model are the same as MHER. We provide empirical results with varying H in Figure 6.

Model-based policy optimization (MBPO) [15] stores short model-generated rollouts branched from real data to the model dataset D_{model} , and optimizes policy using the model dataset D_{model} . For fair comparison, we also utilize DDPG for policy optimization and the size of D_{model} is $1 \cdot 10^6$. Hyper-parameters of DDPG and dynamics model are the same as MHER. The results with model-based horizon $\{3, 5, 7\}$ are reported in Figure 6.

E Comparison with Concurrent Work

We noticed a concurrent work, MapGo [35], which shares similar idea of model-based relabeling (MBR) with us. However, MapGo doesn't exploit the MBR goals deeper. Instead, we further leverage MBR goals for supervised policy learning with theoretical guarantees. Moreover, MapGo utilizes a large number of virtual rollout for policy improvement, and thus introduces model error in both states and actions. In Section 4.2, we have emphasized that MHER avoids training with fully virtual states, and only the goals in the training data are generated by the model. Therefore, our method can be

more robust to model errors. Despite the differences, MapGo found the model cannot fit exactly for hard manipulation tasks, which we have also observed in our experiments and we believe it is worth further research.

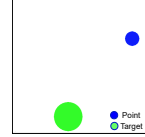
F Task Descriptions

All of the five tasks have continuous state space, action space and goal space. In this section, we will introduce these tasks in detail.

F.1 Point2DLargeEnv-v1

Point2DLargeEnv-v1 is taken from the open-source package *multi-world* and it requires the blue point to reach the green circle. The state space $[-5, 5] \times [-5, 5]$ has two dimensions representing Cartesian coordinates of the blue point, and the action space $[-1, 1] \times [-1, 1]$ also has two dimensions meaning the horizontal and vertical displacement. The goal space is the same as state space, which means $\phi(s) = s$. The blue point and the green circle are randomly initialized in the state space. The allowable error ϵ of reaching goal is the radius of the target circle and is set as 1. The reward function is defined as:

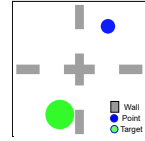
$$r(s_{XY}, a, g_{XY}) = -1(\|s_{XY} - g_{XY}\|_2^2 > \epsilon).$$



Point2DLarge

F.2 Point2D-FourRoom-v1

The Point2D-FourRoom-v1 environment is also built on *multi-world*. The state space, the action space, the goal space, and the reward function are same as Point2DLarge-v1. The difference is that there are four rooms separated by gray walls.

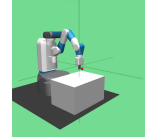


PointFourRoom

F.3 FetchReach-v1

The FetchReach-v1 environment is taken from OpenAI Gym [4]. In this environment, a 7-DoF robotic arm is expected to touch a desired location with its two-finger gripper. The state space is 10-dimensional, including the gripper's position and linear velocities. The action space is 4-dimensional, which represents the gripper's movements and its status about the opening and closing. Moreover, the goals are 3-dimensional vectors representing the target place of the gripper. The state-to-goal mapping is $\phi(s) = s[0 : 3]$, because the first 3 dimensions of state describe the position of the gripper. The allowable error in FetchReach is $\epsilon = 0.05$. The reward function is defined as:

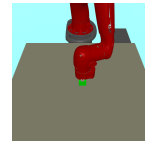
$$r(s_{XYZ}, a, g_{XYZ}) = -1(\|s_{XYZ} - g_{XYZ}\|_2^2 > \epsilon).$$



FetchReach

F.4 SawyerReachXYZEnv-v1

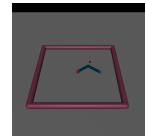
The SawyerReachXYZEnv environment is taken from *multi-world*. The Sawyer robot aims to reach a target position with its end-effector. The observation space is 3-dimensional, representing the 3D Cartesian position of the end-effector. Correspondingly, the goal space is 3-dimensional and describes the expected position, and the state-to-goal mapping is $\phi(s) = s$. Besides, the action space has 3 dimensions describing the next position of the end-effector. The reward function is the same as FetchReach except the allowable error $\epsilon = 0.06$.



SawyerReach

F.5 Reacher-v2

The Reacher environment is revised from OpenAI Gym [4]. States are 11-dimensional, which indicates the angles, the positions, and the velocity of the joints. Actions are 2-dimensional and control the movement of two joints. The



Reacher

goals are 2-dimension representing the expected XY position. And the state-to-goal mapping is $\phi(s) = s[-3 : -1]$, where the last three dimensions are the XYZ position of the end-effect. The reward function is defined as:

$$r(s_{XY}, a, g_{XY}) = -1(\|s_{XY} - g_{XY}\|_2^2 > \epsilon),$$

where the allowable error ϵ is set as 0.02.