Abstract: Meta-reinforcement learning (meta-RL) algorithms allow for agents to learn new behaviors from small amounts of experience, mitigating the sample inefficiency problem in RL. However, while meta-RL agents can adapt quickly to new tasks at test time after experiencing only a few trajectories, the meta-training process is still sample-inefficient. Prior works have found that in the multi-task RL setting, relabeling past transitions and thus sharing experience among tasks can improve sample efficiency and asymptotic performance. We apply this idea to the meta-RL setting and devise a new relabeling method called Hindsight Foresight Relabeling (HFR). We construct a relabeling distribution using the combination of hindsight, which is used to relabel trajectories using reward functions from the training task distribution, and foresight, which takes the relabeled trajectories and computes the utility of each trajectory for each task. HFR is easy to implement and readily compatible with existing meta-RL algorithms. We find that HFR improves performance when compared with existing relabeling methods on a variety of different meta-RL tasks.

Keywords: Reinforcement Learning, Meta-Learning

1 Introduction

Deep Reinforcement Learning (RL) has achieved success on a wide variety of tasks, ranging from computer games to robotics. However, RL agents are typically trained on a single task and are extremely sample-inefficient, often requiring millions of samples to learn a good policy for just that one task. Ideally, RL agents should be able to utilize their prior knowledge and adapt to tasks quickly, just as humans do. Meta-learning, or learning to learn, has achieved promising results in this regard, allowing agents to exploit the shared structure between tasks in order to adapt to new tasks quickly during meta-test time.

However, although meta-learned policies can adapt quickly during meta-test time, training these meta-learned policies still requires a massive amount of experience. Since meta-RL is inherently on-policy due to the assumption that the meta-train time and the meta-test time input distributions should match, many state-of-the-art meta-RL methods [1–5] utilize strictly on-policy data and thus suffer from sample-inefficiency especially during meta-training. Recent works [6, 7] have sought to incorporate off-policy RL [8, 9] into meta-RL to improve sample efficiency.

The combination of off-policy RL and relabeling, in which experience is shared across tasks, has been utilized in the multi-task RL setting, in which an agent learns to achieve multiple different yet related tasks, for both goal-reaching tasks [10] and more general multi-task settings [11]. Experience collected for one task may be completely useless for training a policy to learn that task, but could be extremely informative in training a policy to learn a different task. For example, an agent trying to shoot a hockey puck into a net might miss to the right. This experience could easily be used to train an agent to shoot a puck into a net positioned further to the right [10].

Both meta-RL and multi-task RL involve training on a distribution of tasks, so it follows that we can also combine relabeling techniques with meta-RL algorithms in order to boost both sample efficiency and asymptotic performance. During meta-learning, an agent learns to explore sufficiently to identify

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.
the task it is supposed to be solving, and then uses that knowledge to solve the task. The agent collects exploratory pre-adaptation data, then undergoes some adaptation process using that pre-adaptation data. Finally, after adaptation, the agent attempts to solve the task. In this paper, using concepts from maximum entropy RL (MaxEnt RL), we introduce a relabeling scheme for the meta-RL setting. Prior relabeling methods for multi-task RL have used the total reward of the trajectory under different tasks to guide the relabeling [11, 12]. Direct application of this type of relabeling to the meta-RL setting is potentially sub-optimal since the multi-task RL and meta-RL objectives are distinct (learning to perform many tasks vs. learning to learn a new task). Towards developing an approach more suited to meta-RL, we define the notion of the utility of a trajectory under the different tasks, where the utility captures the usefulness of the trajectory for efficient adaptation under those tasks. We call our method Hindsight Foresight Relabeling (HFR) – we use hindsight in replaying the experience using reward functions from different tasks, and we use foresight in computing the utility of trajectories under different tasks and constructing a relabeling distribution over tasks using these utilities. We demonstrate the efficacy of our method on a variety of robotic manipulation and locomotion tasks. Notably, we show that our method, as the first meta-RL relabeling technique (applied during meta-training) that we are aware of, leads to improved performance compared to prior relabeling schemes designed for multi-task RL.

2 Related Work

Meta-learning, or learning to learn [13–16], has been a topic of interest since the 1980s. Various approaches have been developed in recent years. Prior works have attempted to represent the RL process using a recurrent neural network (RNN) [1, 2, 17] - the hidden state is maintained across episode boundaries and informs the policy as to what task it is currently solving. Similarly, Mishra et al. [4] also maintain the internal state across episode boundaries while incorporating temporal convolution and attention into a recursive architecture. Gradient-based meta-learning methods have also been explored [3, 18–20]. MAML [3] seek to learn a good policy initialization so that only a few gradient steps are needed to achieve good performance on unseen meta-test tasks. Studie et al. [21] build on this gradient-based approach but explicitly consider the effect of the original sampling distribution on final performance. Similar to these works, our relabeling method also considers the impact of pre-adaptation data on the post-adaptation performance. Another body of work focuses on designing strategies for structured exploration in meta-RL such that task-relevant information could be efficiently recovered [6, 22, 23]. Rakelly et al. [6] devise an off-policy meta-RL method called PEARL that trains an encoder to generate a latent context vector on which the meta-RL agent is conditioned. Although we use PEARL as our base algorithm in this work, our relabeling scheme is general enough to be integrated into any meta-RL algorithm.

Sharing experience across tasks has been an understudied problem in the context of meta-RL. Prior work has utilized relabeling at meta-test time to adapt to out-of-distribution tasks [24], but to the best of our knowledge, our paper is the first to incorporate relabeling into the meta-training process. In the context of multi-task RL [25–27], recent methods have proposed relabeling to improve the sample-efficiency [10–12]. HER [10] relabels transitions using goals that the agent actually achieves. Doing so allows for learning even with a sparse binary reward signal. However, HER is only applicable to goal-reaching tasks and cannot be incorporated into meta-RL algorithms because the meta-RL agent is trained on a batch of tasks sampled from a fixed task distribution. Similar to our work, Eysenbach et al. [11] use MaxEnt RL to construct an optimal relabeling distribution for multi-task RL problems and apply this to both goal-reaching tasks and tasks with arbitrary reward functions.

3 Background

3.1 Reinforcement Learning

In reinforcement learning (RL), the environment is modeled as a Markov Decision Process (MDP) \(\mathcal{M} = (S, A, r, p, \gamma, p_1)\), where \(S\) is the state-space, \(A\) is the action-space, \(r\) is the reward function, \(p\) is the transition dynamics, \(\gamma \in [0, 1)\) is the discount factor, and \(p_1\) is the initial state distribution. At timestep \(t\), the agent \(\pi_\theta\), parameterized by parameters \(\theta\), observes the state \(s_t \in S\), takes an action \(a_t \sim \pi_\theta (a_t | s_t)\), and observes the next state \(s_{t+1} \sim p(s_{t+1} | s_t, a_t)\) and the reward \(r(s_t, a_t)\). The goal is to maximize the expected cumulative discounted rewards: \(\max_{\theta} \mathbb{E}_{s_t, a_t \sim \pi_\theta} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \right]\).
3.2 Meta-Reinforcement Learning

In the general meta-reinforcement learning (meta-RL) setting, there is a family of tasks that is characterized by a distribution \( p(\psi) \), where each task \( \psi \) is represented by an MDP \( \mathcal{M}_\psi = (\mathcal{S}, \mathcal{A}, r_\psi, \gamma, p_\psi) \). The tasks share the components \( (\mathcal{S}, \mathcal{A}, \gamma, p_\psi) \), but can differ in the reward function \( r_\psi \) (e.g. navigating to different goal locations) and/or the transition dynamics \( p_\psi \) (e.g. locomotion on different terrains). In this work, we consider the setting where the tasks share the same transition dynamics (i.e., \( p_\psi = p \)), but differ in the reward function. The goal in meta-learning is to learn a set of meta-parameters such that given a new task from \( p(\psi) \) and small amount of data for the new task, the meta-parameters can be efficiently adapted to solve the new task. In the context of meta-RL, given new task \( \psi \), the agent collects some initial trajectories \( \{\tau_{\text{pre}}\} \), each being a sequence \( \{s_1, a_1, s_2, a_2, \ldots \} \), and then undergoes some adaptation procedure \( f_\phi (\pi_\theta, \tau_{\text{pre}}, r_\psi) \) (e.g., a gradient update \cite{Williams1992} or a forward pass through an RNN \cite{Silver2016}). The adaptation procedure returns a new policy \( \pi' \). Using this post-adaptation policy, the agent should seek to maximize the cumulative discounted rewards it achieves. Overall, the meta-RL objective is:

\[
\max_{\theta,\phi} \mathbb{E}_{\psi \sim p(\psi), (s_t, a_t) \sim \pi'(\theta, \phi)} \sum_{t=1}^{\infty} \gamma^{t-1} r_\psi(s_t, a_t) ; \quad \pi'(\theta, \phi) = f_\phi (\pi_\theta, \tau_{\text{pre}}, r_\psi)
\]

where \( \theta, \phi \) are the meta-parameters that are learned in the meta-training phase. A meta-RL agent must learn a good adaptation procedure \( f_\phi \) that is proficient in extracting salient information about the task at hand, using few pre-adaptation trajectories \( \tau_{\text{pre}} \). At the same time, it should learn the policy meta-parameters \( \theta \) such that it can achieve high returns after the adaptation process, i.e., while following the policy \( \pi' \).

3.3 PEARL

In this work, we use PEARL \cite{Chua2018} as our base meta-RL algorithm since it uses off-policy RL and provides structured exploration via posterior sampling. PEARL is built on top of Soft Actor-Critic \cite{Haarnoja2018} and trains an encoder network \( q_\phi (z|c) \) that takes in the “context” \( c \), which consists of a batch of \( (s_t, a_t, r_t) \) transitions, and produces the latent embedding \( z \). The intent is to learn the encoder such that embedding \( z \) encodes some salient information about the task. The adaptation step \( f_\phi \) in PEARL corresponds to generating this latent \( z \) and then conditioning the policy and the value function networks on it. The policy \( \pi_\theta (a|s, z) \) is trained using loss \( L_{\text{actor}} \):

\[
L_{\text{actor}} = \mathbb{E}_{s \sim B, a \sim \pi_\theta, z \sim q_\phi(z|c)} \left[ D_{\text{KL}} \left( \pi_\theta (a|s, z) \| \frac{\exp(Q_\theta(s, a, z))}{Z_\theta(s)} \right) \right]
\]

where \( B \) is the replay buffer. The critic \( Q_\theta (s, a, z) \) and the encoder \( q_\phi (z|c) \) are trained with temporal difference learning:

\[
L_{\text{critic}} = \mathbb{E}_{(s, a, r, s') \sim B, z \sim q_\phi(z|c)} \left[ (Q_\theta (s, a, z) - (r + \tilde{V}(s', \tilde{z})))^2 \right]
\]

where \( \tilde{V} \) is the target state value and \( \tilde{z} \) denotes that the gradient does not flow back through the latent.

4 Hindsight Foresight Relabeling

The objective in this section is to derive a formalism for data-sharing amongst the tasks during the meta-training phase. This is achieved via trajectory-relabeling, wherein a trajectory collected for a training task \( \psi^i \) is reused or re-purposed for training a different task \( \psi^j \). Reward-based trajectory-relabeling has received a lot of attention in recent works on multi-task RL and goal-conditioned RL \cite{Luo2019,Luo2020,Li2021}. The intuition is that if a trajectory \( \tau \) collected while solving for the task \( \psi^i \) achieves high returns under the reward definition for another task \( \psi^j \) (i.e., \( \sum_{t} r_{\psi^j}(s_t, a_t) \) is large), then \( \tau \) can be readily used for policy-optimization for the task \( \psi^j \) as well. The meta-RL setting presents the following subtlety – for any given task, the meta-RL agent generates trajectories with the aim of utilizing them in the adaptation procedure and subsequently seeks to maximize the post-adaptation returns (c.f. §3.2). To improve the efficiency of the meta-training stage, we would like to share these pre-adaptation trajectories amongst the different tasks, accounting for the fact that the metric of
interest with these trajectories is their usefulness for task-identification, rather than the returns (as in multi-task RL). This difference is illustrated in Figure 3. Hence, when deciding if a trajectory \( \tau \) collected for task \( \psi^i \) is appropriate to be reused for task \( \psi^j \), it is sub-optimal to consider the return value of this trajectory under \( \psi^j \). Instead, we argue that this reuse compatibility should be determined based on the performance on the task \( \psi^j \), after the agent has undergone adaptation using \( \tau \). Concretely, we define a function to measure the utility of the trajectory \( \tau \) for a task \( \psi^j \):

\[
U_{\psi^j}(\tau) = \mathbb{E}_{s_t, a_t \sim \pi’} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} r_{\psi^j}(s_t, a_t) \right]
\]

(4)

where \( \pi’ = f_{\phi} (\pi_\theta, \tau, r_{\psi^j}) \) denotes the policy after using \( \tau \) for adaptation. The trajectory-relabeling mechanism during meta-training now incorporates this function \( U_{\psi^j} \), which we refer to as the utility function, rather than the return \( R_{\psi^j} \). Broadly, a trajectory \( \tau \) collected for task \( \psi^i \) can be relabeled for use in another task \( \psi^j \) if \( U_{\psi^j}(\tau) \) is high. Subsection §4.1 makes this more precise by deriving a relabeling distribution \( q(\psi | \tau) \) that informs us of the tasks for which \( \tau \) should be reused. Figure 4, and the caption therein, describe a high-level overview of our approach, HFR.

Comparison to HIPI [11] with a didactic example. We consider a toy environment to further motivate that return-value based data sharing and trajectory relabeling [11] is potentially sub-optimal for meta-RL. The Four-Corners environment consists of a point robot placed at the center of a square where each corner of the square represents a goal location, as shown in Figure 1. For each goal (task), there is a section of the space in the corresponding quadrant in which the robot receives a large negative reward. Consider a trajectory \( \tau \) that hovers over the blue square in top-right quadrant. Note that \( \tau \) could have been generated by the meta-RL agent while training on any of the four tasks. We examine if \( \tau \) can be reused for the blue task. Since \( R_{\text{blue}}(\tau) \) is highly negative, the relabeling approach in HIPI does not reuse \( \tau \) for training the meta-RL agent on the blue task. It is easy to see, however, that \( \tau \) carries a significant amount of signal pertaining to task-identification on the blue task, making it a useful pre-adaptation trajectory. HFR reuses \( \tau \) for the blue task since the utility \( U_{\text{blue}}(\tau) \) is high. Figure 2 compares HFR and HIPI in terms of the success-rate in the Four-Corners environment, and shows the quantitative benefit of using the utility function for trajectory relabeling.

4.1 Deriving a Meta-RL Relabeling Distribution

Our derivation in this subsection largely follows HIPI [11], but differs in that we adapt it to the meta-RL setting to promote sharing of pre-adaptation trajectories amongst tasks, using the concept of trajectory utility. Assume a dataset \( D \) of trajectories gathered by the meta-RL agent when solving the different tasks in the meta-train task distribution. We wish to learn a trajectory relabeling distribution \( q(\psi | \tau) \) such that, given any trajectory \( \tau \sim D \), we could reuse \( \tau \) for tasks with high density under this posterior distribution. To that end, we start by defining a variational distribution \( q(\tau | \psi) \) to designate the trajectories used for the adaptation process \( f_{\phi} \), for a given task \( \psi \). Using the definition of the utility function (Eq. 4), the meta-RL objective from Eq. 1 could be written as:

\[
\max_{\theta, \phi} \mathbb{E}_{\psi \sim p(\psi)} \mathbb{E}_{\tau \sim q(\tau | \psi)} [U_{\psi}(\tau)]
\]

For fixed meta-parameters \((\theta, \phi)\), a natural approach to optimize
Figure 4: Consider a trajectory $\tau(\psi)$ that is collected for the task $\psi$. HFR uses hindsight to relabel this trajectory using reward functions for different tasks, and then uses foresight to compute the utility of the relabeled trajectory for the different tasks. A distribution over tasks is constructed using the utilities, and a task $\psi$ is sampled from the distribution, with tasks for which the trajectory has higher (normalized) utility having higher probability mass. The trajectory is then relabeled using the reward function $r_{\psi}$ and added to the task-specific replay buffer $B_{\psi}$. HFR uses PEARL as the base meta-RL algorithm. Shown in this figure are the encoder $q_\phi(z|c)$ and the $z$-conditioned policy and Q-value networks (c.f. §3.3). HFR does not alter the meta-training update rules used by PEARL.

the variational distribution $q(\tau|\psi)$ is to use this same objective since it facilitates alignment with thegoals of the meta-learner. Thus, the combined objective for the variational distributions for all thetasks, augmented with entropy regularization, is:

$$\max_q \mathbb{E}_{\psi \sim p(\psi)} \left[ \mathbb{E}_{\tau \sim q(\tau|\psi)} [U_\psi(\tau)] + H_q(\tau|\psi) \right]$$

where $H_q(\tau|\psi)$ denotes the causal entropy of the policy associated with $q(\tau|\psi)$. Now, it is easy toshow that the above optimization is equivalent to a reverse-KL divergence minimization objective:

$$\min_{q(\tau,\psi)} D_{\text{KL}}[ q(\tau,\psi) \| p(\tau,\psi) ]$$

In this objective, the joint distributions over the tasks and thetrajectories are defined as $q(\tau,\psi) = q(\tau|\psi)p(\psi)$ and $p(\tau,\psi) = p(\tau|\psi)p(\psi)$, where

$$p(\tau|\psi) = \frac{1}{Z(\psi)p_1(s_1)}e^{U_\psi(\tau)} \prod_{t=1}^T p(s_{t+1}|s_t, a_t)$$

Our goal is to formulate the trajectory relabeling distribution $q(\psi|\tau)$. To make this explicit in ourobjective, we use the trick proposed in HIPI [11] and factor $q(\tau,\psi)$ as $q(\psi|\tau)q(\tau)$, thereby rewritingthe reverse-KL divergence minimization objective as:

$$\min_{q(\tau,\psi)} \mathbb{E}_{\tau \sim q(\tau)} \left[ \log q(\psi|\tau) + \log q(\tau) - \log p(\psi) + \log Z(\psi) - U_\psi(\tau) - \log p_1(s_1) - \sum_{t} \log p(s_{t+1}|s_t, a_t) \right]$$

Ignoring the terms independent of $\psi$, we can analytically solve (by differentiating and setting to zero)for the optimal trajectory relabeling distribution for the meta-RL setting:

$$q(\psi|\tau) \propto p(\psi)e^{U_\psi(\tau) - \log Z(\psi)} \quad (5)$$

Given a trajectory $\tau \sim D$, the HFR algorithm uses this relabeling distribution to sample tasks forwhich $\tau$ should be reused. Concretely, we compute the utility function under $\tau$ for all the tasks,construct the distribution $q(\psi|\tau)$ using these utilities (Eq. 5), and sample tasks from it. Please seeFigure 4 for details. We assume a uniform prior $p(\psi)$ over the tasks in our experiments.

4.2 Algorithm and Implementation details

Our relabeling algorithm is summarized in Algorithm 1 and fits seamlessly into any of the basemeta-RL algorithms. Once the meta-RL agent generates a trajectory $\tau$ for a training task, $\tau$ is fed asinput to HFR, and it returns another task that can reuse this experience $\tau$. We compute the utility ofthe input trajectory for every task, along with an empirical estimate of the log-partition function ofthe tasks. The task to relabel the trajectory with is then sampled from a categorical distribution. For
We highlight that HFR facilitates efficient data-sharing among the tasks via trajectory-relabeling.

Algorithm 1:

Input: Trajectory to be relabeled (τ)
Output: Task to relabel the trajectory with (ψ)

for each task ψ' do
    \( U_{ψ'}(τ) \leftarrow \text{ComputeUtility}(τ, ψ') \)
    \( \log Z(ψ') \leftarrow \text{GetLogPartition}(ψ') \)
end

Return \( ψ \sim \text{softmax}\{U_{ψ'}(τ) - \log Z(ψ')\} \) (Eq. 5)

Function GetLogPartition(ψ):

Sample batch of trajectories \( \{τ^i\}_{i=1}^N \sim B_ψ \)
for each trajectory \( τ^i \) do
    \( U_ψ(τ^i) \leftarrow \text{ComputeUtility}(τ^i, ψ) \)
end

Return \( \log Z(ψ) \approx \log \left( \frac{1}{N} \sum_{i=1}^N e^{U_ψ(τ^i)} \right) \)

Function ComputeUtility(τ, ψ):

for each \((s_t, a_t, r_t) \in τ\) do
    Replace \( r_t \) with \( r_ψ(s_t, a_t) \)
end

Sample embedding using encoder \( z \sim q_φ(z|τ) \)
Sample a batch of initial states \( \{s'_1\}_{i=1}^N \sim B_ψ \)
Sample actions for these states using the post-adaptation policy \( π_θ(s'_i | s, z) \):
\( \{a_1 \sim π_θ(a_1 | s'_1, z)\}_{i=1}^N \)

Return \( U_ψ = \frac{1}{N} \sum_{i=1}^N Q_θ(s'_1, a'_1, z) \) (Eq. 6)

our experiments, we build on top of the PEARL algorithm [6], which is a data-efficient off-policy meta-RL method. PEARL maintains task-specific replay buffers \( B_{ψ'} \). If HFR returns the task \( ψ' \), then \( τ \) is relabeled using the reward function \( r_ψ \) and added to \( B_{ψ'} \) for meta-training on the task \( ψ' \).

The adaptation procedure \( π' = f_ψ(π_θ, τ, r_ψ) \) for a task \( ψ \) corresponds to a sequence of steps: 1.) augment \( τ \) by marking each transition with a reward value computed using \( r_ψ(s_t, a_t) \), 2.) condition the encoder on \( τ \) to sample an embedding, \( z \sim q_φ(z|τ) \); and 3.) condition the policy on \( z \) to obtain the post-adaptation policy, \( π' = π_θ(·|s, z) \). The calculation of the utility function (Eq. 4) requires generation of post-adaptation trajectories, which could be computationally inefficient, especially if the number of tasks is large. To avoid this cost, for each task, we sample a batch of initial states \( s_1 \sim p_1(s_1) \) and the corresponding actions from the post-adaptation policy, and compute the utility based on an estimate of the state-action value function \( Q_ψ''(s_1, a_1) \) as:

\[
U_ψ(τ) = E_{s_1 \sim p_1, a_1 \sim π'|s_1} \left[ Q_ψ''(s_1, a_1) \right]
\] (6)

Since we use PEARL, we can avoid training separate task-specific value functions \( Q_ψ \), and instead get the required estimates from the task-conditioned critic \( Q(s, a, z) \) already used by PEARL (Eq. 3).

We highlight that HFR facilitates efficient data-sharing among the tasks via trajectory-relabeling without altering the meta-train and test-time adaptation rules of the base meta-RL algorithm.

5 Experiments

The goal in this section is to quantitatively evaluate the benefit of sharing experience among tasks using HFR, during the meta-train stage. We evaluate on a set of both sparse and dense reward MuJoCo environments [28] modeled in OpenAI Gym [29]. Please refer to the Appendix for environment details. We compare HFR with two relabeling methods: Random, in which each trajectory is relabeled with a randomly chosen task, and HIPI [11], which utilizes MaxEnt RL to devise an algorithm that relabels each transition using a distribution over tasks involving the soft Q values for that transition. In contrast, HFR proposes the concept of the utility of a trajectory for a given task. Using the utility aligns the relabeling methodology with the objective of the meta-RL agent (c.f. section 4). All methods are built on top of PEARL. Finally, we compare to PEARL with no relabeling at all, which we refer to as None.

5.1 Results

Sparse Reward Environments: We first evaluate HFR on a set of sparse reward goal-reaching environments. We use four environments: a goal-reaching task involving a quadruped Ant robot, a pushing task on the Sawyer robot, a reaching task on the Sawyer robot, and a velocity-matching task involving a bipedal Cheetah robot. Figure 6 plots the results and shows that our method learns more quickly than the baselines and achieves higher asymptotic returns/success rates. Meta-RL
in sparse reward tasks is hard due to the challenges of task-identification and efficient exploration. HFR is especially useful for these tasks as the data-sharing afforded by the trajectory relabeling algorithm mitigates the need for an elaborate exploration strategy during meta-training. This leads to the sample-efficiency gains exhibited in Figure 6.

**Dense Reward Environments:** We next evaluate HFR on dense reward environments (Figure 7). We experiment with the Cheetah-Highdim environment, in which the bipedal robot is required to best match its state vector to a set of predefined vectors, as well as the Cheetah-Vel and quadruped Ant-Vel environments, in which the robots are required to run at various velocities.

Note that the impact of HFR is much less pronounced for these environments, with the exception of Cheetah-Highdim. We believe this is because exploration is not as critical for these environments as it was for the sparse reward tasks. This hypothesis is supported by the fact that, in these environments, PEARL with no relabeling is competitive with the various relabeling methods, which all share similar performance, whereas in the sparse reward environments HFR is the relabeling method that performs best, with the two other relabeling methods also vastly outperforming vanilla PEARL. In the case of the Cheetah-Vel and Ant-Vel environments, agents are provided with an informative dense reward that immediately informs them as to which task they’re supposed to be solving. Although the agent in Cheetah-Highdim is also provided with an informative dense reward, the reward function in this task is a linear combination of an 18-dimensional state vector. Thus, reasonably good exploration is needed to determine optimal values for each of these 18 dimensions. HFR relabeling provides improvement over the baselines for this task.

**5.2 Ablation Studies**

**Batch Size:** We investigate the impact of the batch size $N$ used in Algorithm 1 for computing an empirical estimate of the state-action value: $\frac{1}{N} \sum_{i=1}^{N} Q_\theta (s_i, a_i, z)$. We compare the effect of batch size across batch sizes \{16, 32, 64, 128, 256\}. We expect lower values of $N$ to lead to a higher variance estimate of the post-adaptation cumulative discounted rewards and thus potentially a worse approximation of the optimal meta-RL relabeling distribution. In Figure 8a, we see some evidence of this in the comparatively worse performance when using $N = 16$ and $N = 32$. However, we note that even with these small batch sizes, our method still performs well, and in general achieves high returns across all choices of $N$.

**Partition Function:** We investigate the impact of the log-partition function $\log Z(\psi)$ in the optimal meta-RL relabeling distribution (Eq. 5). Prior work has noted the importance of the partition function
in the multi-task RL setting when tasks may have different reward scales [11]. We believe that in the meta-RL setting, the partition function may be crucial even if the tasks share the reward scale, since some tasks in the meta-training distribution may be easier to solve than others. We speculate that with the omission of the partition function from our relabeling distribution, trajectories would find high utility and thus be disproportionately labeled with the easily-solved tasks, causing a degradation in overall performance. In our experiments, we find that the partition function serves as an essential normalization factor; Figure 8b shows an example.

**Reward Function:** One assumption our method assumes is access to the true reward function \( r(\psi) \), which we can query to get the reward for an individual transition under any task \( \psi \). This availability has also been utilized in existing works on relabeling for multi-task RL [11, 12]. In many real-world applications of meta-RL, e.g., a distribution over robotic tasks, it is reasonable to assume that the task-designer outlines a rough template for the rewards corresponding to the different tasks from the distribution. Furthermore, several of the rewards used in our experiments are success/failure indicators, which are simple to specify. Nevertheless, we consider the scenario where we cannot query the true reward function for individual transitions. Figures 8c and 8d show good performance even when we use a learned reward function rather than the true reward function to relabel trajectories.

### 6 Conclusion

In this paper, we introduced HFR, a trajectory relabeling method for meta-RL that enables data sharing between tasks during meta-train. We argue that unlike the multi-task RL setting, where the appropriateness of a trajectory for a task could be measured by the returns under the task, for meta-RL, it is preferable to consider the future (expected) task-returns of an agent adapted using that trajectory. We capture this notion by defining the utility function for a trajectory-task pair and incorporate these utilities in our relabeling mechanism. Inspired by prior work on multi-task RL, an optimal relabeling distribution is then derived that informs us of the tasks for which a generated trajectory should be reused. Hindsight is used to relabel trajectories with different reward functions, while foresight is used in computing the utility of each trajectory under different tasks and constructing a relabeling distribution. HFR is easy to implement, can be integrated into any existing meta-RL algorithm, and yields improvement on a variety of meta-RL tasks, especially those with sparse rewards. To the best of our knowledge, HFR is the first relabeling method designed explicitly for the meta-RL paradigm.
References

[1] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[2] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[3] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[4] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[5] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel. Prompt: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

[6] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340, 2019.

[7] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola. Meta-q-learning. *arXiv preprint arXiv:1910.00125*, 2019.

[8] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[9] S. Fujimoto, H. Van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[10] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, pages 5048–5058, 2017.

[11] B. Eysenbach, X. Geng, S. Levine, and R. Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *arXiv preprint arXiv:2002.11089*, 2020.

[12] A. C. Li, L. Pinto, and P. Abbeel. Generalized hindsight for reinforcement learning. *arXiv preprint arXiv:2002.11708*, 2020.

[13] J. Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta... hook.* Diploma thesis, Institut f. Informatik, Tech. Univ. Munich, 1(2), 1987.

[14] D. Naik, R. Mammone, and A. Agarwal. Meta-neural network approach to learning by learning. In *Proceedings of the 1992 Artificial Neural Networks in Engineering, ANNIE’92*, 1992.

[15] S. Thrun and L. Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[16] J. Baxter. Theoretical models of learning to learn. In *Learning to learn*, pages 71–94. Springer, 1998.

[17] T. Miconi, K. Stanley, and J. Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pages 3559–3568. PMLR, 2018.

[18] A. Nichol and J. Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(2):1, 2018.

[19] Z. Xu, H. van Hasselt, and D. Silver. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*, 2018.

[20] Z. Zheng, J. Oh, M. Hessel, Z. Xu, M. Kroiss, H. Van Hasselt, D. Silver, and S. Singh. What can learned intrinsic rewards capture? In *International Conference on Machine Learning*, pages 11436–11446. PMLR, 2020.
[21] B. C. Stadie, G. Yang, R. Houthooft, X. Chen, Y. Duan, Y. Wu, P. Abbeel, and I. Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

[22] L. Zintgraf, K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

[23] E. Z. Liu, A. Raghunathan, P. Liang, and C. Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. *arXiv preprint arXiv:2008.02790*, 2020.

[24] R. Mendonca, X. Geng, C. Finn, and S. Levine. Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling. *arXiv preprint arXiv:2006.07178*, 2020.

[25] L. P. Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[26] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[27] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

[28] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[29] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.