Hindsight Planner

Yaqing Lai
Tsinghua University
Beijing, P.R.China
yastingl@gmail.com

Wufan Wang
Tsinghua University
Beijing, P.R.China
answer_wwf@163.com

Yunjie Yang
Tsinghua University
Beijing, P.R.China
yyj15@mails.tsinghua.edu.cn

Jihong Zhu
Tsinghua University
Beijing, P.R.China
Jihong_Zhu@hotmail.com

Minchi Kuang
Tsinghua University
Beijing, P.R.China
kuangminchi@sina.com

ABSTRACT

Goal-oriented reinforcement learning capacitates agents to accomplish variant goals, which is crucial for robotic tasks. However, the sparse-reward setting of these tasks aggravates sample inefficiency. Hindsight Experience Replay (HER) was introduced as a technique to elevate sample efficiency by imaging hindsight virtual goals for unsuccessful trajectories, which mitigates long-term domination of negative rewards. Nevertheless, there is still a gap between the distribution of hindsight goals and desired goals of the tasks, which was narrowed by lots of aimless exploration in HER. In this paper, we propose Hindsight Planner(HP) to generate several subgoals guiding the agent to explore towards the desired goal step by step, which allows the agent to exploit its local knowledge learned from achieved goals. The planner uses history trajectories to learn the structure of feasible goal space, then generalizes its knowledge to unseen goals. We have extensively evaluated our framework on a number of robotic tasks and show substantial improvements over the original HER in terms of sample efficiency and converged performance.

KEYWORDS

deep reinforcement learning; goal-oriented learning; task planning

ACM Reference Format:
Yaqing Lai, Wufan Wang, Yunjie Yang, Jihong Zhu, and Minchi Kuang. 2020. Hindsight Planner. In *Proc. of the 19th International Conference on Autonomous and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Recent progress in deep reinforcement learning (RL), including value-based methods and policy gradient methods, have continued to push the boundaries of the ability of RL agents, from playing games [13, 25] to solving robotic tasks [6, 20, 30].

Despite its notable success, there still exists a common challenge that we need to design a proper reward function to guide the policy learning and avoid unexpected hacking behavior with careful shaping [15, 19], sometimes leading to numerous experiments. The substantial task-specific knowledge and RL expertise required in this process hinders the application of RL methods to broader fields, including real-world problems. Therefore, the natural and convenient sparse reward setting attracts more and more researchers' attention. Moreover, many previous efforts proved that sparse indicator rewards, rather than the shaped dense rewards, often induce more rational behaviors close to what the designer wants [1, 30]. However, the sparse reward setting presents another challenge for current deep RL algorithms: sparse positive signals deteriorate the sample inefficiency with which RL methods have been confronting all the time. Therefore, an important direction for RL research is towards more sample efficient methods that reduce the number of environment interactions, yet can be trained using only sparse rewards.

To this end, Andrychowicz et al. [1] introduced the idea of Hindsight Experience Replay(HER), which can efficiently train a goal-conditioned policy by retroactively transforming failed trajectories into successful ones. HER makes use of failed attempts by re-labeling original goals with achieved goals mapping from visited states and constructing new positive transitions into replay buffer, which can be seen as a form of implicit curriculum learning. However, the distribution of hindsight goals differs from that of task goals, which has to be approximated via a large number of aimless explorations.

In this work, we address the limitation by introducing a method called Hindsight Planner (HP). We propose this technique based on the assumption that after some explorations around the initial state, agents have acquired enough knowledge about the local space around it. It is still difficult to reach faraway goals. We hope to learn a planner that knows how to break down inapproachable goals to a series of subgoals which are much easier to get close to with the knowledge that agents already have.

For the purpose of not demanding extra interactions with environment and the generality of algorithm, we propose to take advantage of the goal sequences that collected trajectories have explored and verified to provide data for planner learning. In other words, the achieved goal mapping from the terminal state of one trajectory could be approached by that goal sequence (a goal sequence is the projection of a trajectory into goal space). That goal sequence could tell us how can a goal be achieved, and then help us to infer goal sequences for unseen goals. In this way, rather than by the slow spreading of state values or state-action values, guiding an agent to act towards a given goal could be much easier and faster. Hence, the distribution of hindsight goals is accelerated to approximate to task goals distribution.

Proc. of the 19th International Conference on Autonomous and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

In order to learn such a generator of goal sequences, or to be more practical, subsequences, we use the supervised learning method to train a planner. To be specific, we choose some points of that goal sequence as the supervising label of subgoals which the planner needs to produce given starting point and ultimate goal. To pick out waypoints that are crucial for reaching the terminal goal, we present two simple but effective strategies: time-sample and route-sample.

In summary, we introduce a method to accelerate the approximation of hindsight goals distribution to task goals distribution in HER, meanwhile providing the property of general applicability. Beyond combination with HER, our method is also compatible with other RL algorithms, including off-policy and on-policy ones, due to its minimal assumption about the RL algorithm and specific tasks. We evaluate our method on three kinds of robotic environments. From the experiment results, comparing to vanilla HER and another technique made to improve HER, we observe superiorities of our method, in terms of both sample efficiency and converged performance.

2 BACKGROUND

In this section, we present introductions to those relevant concepts for goal-oriented reinforcement learning with sparse reward as well as RL algorithms used in this paper.

2.1 Goal-oriented Reinforcement Learning

Reinforcement learning pursues the goal of finding the policy π for an agent interacting with an environment which maximizes the expected reward. In the traditional RL framework, we assume that the environment is fully observable, thus the interacting can be modeled as a Markov Decision Process(MDP), involving the the environment state $s \in S$ and agent action $a \in A$, with the transition probabilities $p(s_{t+1}|s_t, a_t)$ and reward function $r : S \times A \longrightarrow \mathbb{R}$ that yields a reward for the action a_t performed over state s_t, and also a discount factor $\gamma \in [0, 1]$. The policy is the mapping from a state to an action: $\pi : S \longrightarrow A$.

We consider goal-oriented reinforcement learning with sparse rewards setting. Universal Value Function Approximators [22] was introduced to solve goal-oriented MDP. Let G be the goal space. Afterwards, the reward function adapts to condition on a given goal $g \in G$, such that $r^g : S \times A \times G \longrightarrow \mathbb{R}$. Every episode starts with sampling a state-goal from some distribution $p(s_0, g)$, while the goal g stays fixed during the whole episode. At each timestep, the agent also takes the goal into consideration $\pi : S \times G \longrightarrow A$ and gets the rewards r_t^g. In the sparse reward setting, the reward function usually has the following form:

$$r_t^g = r(s_t, a_t, s_{t+1}, g) = \begin{cases} 0, & if \ |\psi(s_{t+1}) - g| \le \delta_g \\ -1, & otherwise \end{cases} \tag{1}$$

while $\psi : S \longrightarrow G$ is a given function that projects a state into goal space G. δ_g is a predefined threshold that indicating whether the goal is considered to be reached (see [18]). $|\cdot|$ is a distance matric.

As stated before, we model the policy as a conditional probability distribution over state and goal, $\pi_\theta(a_t|s_t, g)$, where θ are learnable parameters. Following policy gradient methods, Our objective is to optimize θ with respect to the expected cumulative reward, given by:

$$J(\theta) = \mathbb{E}_{s_0=s, a_t\sim\pi(\cdot|s_t,g), s_{t+1}\sim p(s_{t+1}|s_t,a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}, g)\right] \tag{2}$$

To simplify the notation, we simply use $\mathbb{E}_\pi[\cdot]$ to denote expectation $\mathbb{E}_{s_0=s, a_t\sim\pi(\cdot|s_t,g), s_{t+1}\sim p(s_{t+1}|s_t,a_t)}[\cdot]$ in the rest of paper. According to the policy gradient theorem (Sutton et al. [28]), the gradient of $J(\theta)$ can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi[\nabla log\pi(a|s, g) Q^\pi(s, a, g)] \tag{3}$$

where $Q^\pi(s, a, g) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}, g)]$, called the critic, denotes the expected return under poliy π after taking an action in state s, with goal g.

2.2 Deep Deterministic Policy Gradient

The objective $J(\theta)$ could be maximized using value function methods, policy gradient, or the combination of both, i.e. the actor-critic methods. For continuous control tasks, Deep Deterministic Policy Gradient (DDPG) [9] shows promising performance, which is essentially an off-policy actor-critic method. However, our proposed framework is not restricted to DDPG. There are two separate neural networks carry out the role of actor $\mu(s, g)$ and critic $Q(s, a, g)$. The agent uses a behavior policy to explore the environment, whose action is sampled from actor network plus some noise, $a \sim \mu(s, g) + \mathcal{N}(0, 1)$. The critic is trained using temporal difference with the target $y_t = r_t + \gamma Q(s_{t+1}, g_t, \mu(s_{t+1}, g_t))$. The actor is trained using policy gradient by descending on the gradients of loss $\mathcal{L}_a = -J(\theta) = -\mathbb{E}_s[Q(s, g, \mu(s, g))]$ through the deterministic policy gradient algorithms [24],

$$\nabla J(\theta) = \mathbb{E}[\nabla_\theta \mu(s, g) \nabla_a Q(s, g, a)|_{a=\mu(s,g)}] \tag{4}$$

Out of consideration for training stability, two additional networks are maintained as the target networks, whose weights are periodically updated to the weights of latest network [9, 12, 13].

2.3 Hindsight Experience Replay

Learning with the presence of sparse rewards which save the annoyance of reward shaping remains a challenge. Intuitively, the reason why learning from sparse reward settings has the problem of inefficiency is that the extremely low probability of encountering a given goal leads to the absolute domination of negative signals, therefore the gradient method is difficult to choose effective promoting direction based on minuscular positive transitions. Hindsight Experience Replay (HER) [1] is a remarkable advance to address this challenge by exploiting failure trajectories in an alternative way. The main idea of HER is substituting the original goal g by what it actually achieved (g') during experience replay as if the agent was instructed to reach g' from the beginning, which greatly increases the ratio of positive transitions feed into gradient computing, thus make the learning more viable. In practice, the $future$ strategy in HER works well by randomly sampling a mini-batch of episodes from buffer. Then for each episode $\{\{s_i\}_{i=1}^T, \{a_i\}_{i=1}^T, \{g_i\}_{i=1}^T, \{r_i\}_{i=1}^T, \{s_i'\}_{i=1}^T\}$, it randomly chooses $s_j, s_k \in \{s_i\}_{i=1}^T, 1 \le j < k \le T$ and relabels transition $(s_j, a_j, g_j, r_j, s_{j+1})$ to $(s_j, a_j, \psi(s_k), r_j', s_{j+1})$, where r_j' is recomputed according to Eq.1.

3 HINDSIGHT PLANNER

3.1 Motivation

As stated above, HER proposed a highly effective scheme to tackle the challenge of sparse reward signals in robotic manipulation tasks. However, the replacement of desired goals by hindsight virtual goals shifts the task distribution towards the near-at-hand tasks, which does not provide the guarantee of accomplishment of original tasks. In other words, unless the distribution of hindsight goals is close to that of original goals or approximates to it through exploration gradually, the agent will keep performing badly in terms of original goals. In HER, the distribution gap was narrowed down by aimless exploration, although it can get much local information around its initial states. Both [1] and [21] have observed that if the original goal was restricted to a small enough area, the learning became inefficient again and easy to get stuck in the area around the starting point. These phenomena suggest the inability of generalization to strange goals from hindsight goals.

However, as humans, we know how to reach faraway goals even if we only have skills of moving around locally: break down the far goals into several subgoals which can be accomplished by our local knowledge one by one. Of course, our local knowledge around each subgoal is not acquired at a time, but the arrival at the subgoal enables us to explore the space closer to the ultimate goal. That is to say, the planned subgoals allow us to explore with higher intentionality.

Take an example for toy ant locomotion on a plain, assuming that the ant has known how to move to any goal around it within one decimetre (by hindsight experience replay). Right now, we give it a goal five decimetres away, which is still bizarre to the ant of current level. The ant might know the rough direction to take action, but not precise and confident. If we make the assumptions that the opening angle of direction the ant might move in is proportional to the distance between its state and goal, and that the movement direction keeps unchanged over a period of time. Then, given the red dot as ultimate goal, the possible areas in which the ant might explore could be represented by the blue shaded fans in Figure 1. However, if the ultimate goal were broken down to three subgoals (yellow dots), the opening angle is substantially narrowed, depicted by yellow shaded fans, which enables the exploration towards the ultimate goal meaningful.

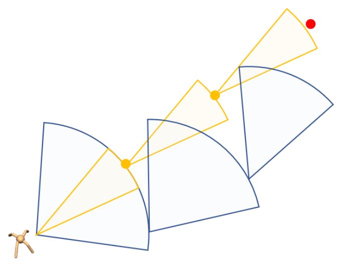


Figure 1: A far goal confuses the ant into bigger exploration area, while subgoals narrow down the exploration by exploiting the principle of locality. Red dot is the ultimate goal and yellow dots are subgoals.

But what helps us to do the breaking down? The understanding of legal goal space structure and environment. Well, the next question would be "How to capture the structure of goal space?". In fact, we already have the basic knowledge about the structure of goal space after our preliminary exploration. Further, the knowledge about the goal sequence leading to task goals would be more and more precise as we perform further intentional exploration.

Based on these discussions, we propose hindsight planner combining with HER to accelerate the learning process and lift the task completion capability.

3.2 Algorithm Framework

3.2.1 Planner Design. The idea behind Hindsight Planner(HP) is very straightforward: after experiencing some episodes $\tau^0, \tau^1, ..., \tau^m$ which terminates at $s_T^0, s_T^1, ..., s_T^m$, we now already have practical trajectories to instruct the planner to generate subgoals. To be specific, given the number of subgoals k, we pick out k most critical waypoints from each episode $s_{i_1}, s_{i_2}, ..., s_{i_k}, 0 \leq i \leq m$, the goals achieved at these k waypoints, $g_{i_1}, g_{i_1}, .., g_{i_k}$, are what the planner needs to output given starting point and destination. For the simplicity of notation, we just use the $g_1, ..., g_k$ to refer to $g_{i_1}, ..., g_{i_k}$ where ambiguity would not be incurred. To train the planner, we set the learning objective as maximizing the likelihood:

$$\max P(\boldsymbol{g}|g_s, g_u) = P(g_1, g_2, ..., g_k|g_s, g_u)$$

Now, we need to model the $P(g_1, g_2, ..., g_k|g_s, g_u)$. Note that when we, as humans, make plans, we usually break up the ultimate goal into the first subgoal, then the second subgoal, and so forth (of course, it is also possible to draw up the last subgoal firstly, then the second last one, and infer backward to the first one. Since they have similar formation, we just discuss the former situation). That is to say, the first subgoal is conditioned on (g_s, g_u), the second one is conditioned on $(g_1; g_s, g_u)$, the last one g_k is conditioned on $(g_{k-1}, ..., g_1; g_s, g_u)$. Thus, the joint probability of $\boldsymbol{g}$ conditioned on (g_s, g_u) would be $P(g_1|g_s, g_u)\Pi_{i=2}^{k} P(g_i|g_{i-1}, ..., g_1; g_s, g_u)$. We use the logarithmic form of the objective for the consideration of practical optimization, and designate the parameterized planner with ϕ, which gives the following form of objective:

$$\max_{\phi} logP_{\phi}(\boldsymbol{g}|g_s, g_u) = logP_{\phi}(g_1|g_s, g_u) + \sum_{i=2}^{k} logP_{\phi}(g_i|g_{i-1}, ..., g_1; g_s, g_u) \tag{5}$$

Under the settings of finite buffer B of trajectories, we maximize the summation of logarithmic probability with respect to each trajectory $\tau \in B$. Furthermore, we may prioritize the trajectories that have the properties we appreciate, such as fast completion, stable control of robotic components, and etc. The relative worse trajectories may include some unnecessary motion due to the exploration factor or the limited level of current policy. Therefore, we multiply the $P(g|g_s, g_u)$ by weighting factor ρ, which evaluating the quality of that trajectory according to its start and end. In the ant locomotion example, ρ would be the distance between start and termination divided by completion time, which depicts the agility of ant. Hence, the prioritized objective is given by:

$$\max_{\phi} \sum_{\tau \in B} logP_{\phi}(\boldsymbol{g}^{\tau} | g_s^{\tau}, g_u^{\tau}) \cdot \rho(\tau) \quad (6)$$

Actually, the formation of our objective resembles that of translation tasks, where the researchers usually approach the translation problem by maximizing the probability of sentence in one language translated from source sentence in another language by human. In this domain, the sequence-to-sequence(seq2seq) models show extraordinary capability in recent years [27, 32]. We adopt the seq2seq thoughts and the basic recurrent neural network cell LSTM to constitute our planner architecture. From the perspective of seq2seq, the source sequence contains the starting point g_s and ultimate goal g_u, while k subgoals act as the role of target sequence. In fact, we regard the destination as the end-of-sentence "<EOS>", which is used in translation task broadly, to indicate the end of source sequence and the start of the process of predicting target sequence, saving the introduction of an extra meaningless vector. Therefore, strictly speaking, the source sequence only contains g_s in our model. Since the source sequence in our task is quite short, we do not introduce the attention technique into our architecture. As shown in Figure 2, the model takes the g_s as the input of encoder and the ultimate g_u as the first input of decoder, then produces subgoals one by one. In addition, we also experimented with the architecture that produces subgoals in reverse order, showing insignificant differences in terms of learning speed and performance.

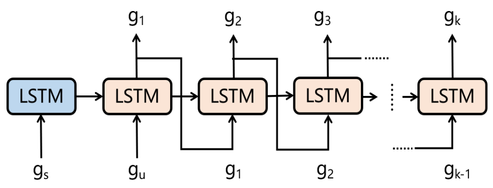


Figure 2: Planner architecture

3.2.2 Hindsight Waypoints Sampling Strategy. One choice which has to be made in order to use HP is the picking of critical waypoints used for SL. For generality, we intend to make as few assumptions as possible about the specific tasks, therefore we proposed two simple and general strategies, called *time-sample* and *route-sample*. As their names imply, they select waypoints with equal interval of time and equal interval of route length.

(1) **Time-sample**. Consider the total timestep of one trajectory is T, we pick the waypoints on the trajectory with interval of $\frac{T}{k}$ timestep.
(2) **Route-sample**. We take the route of trajectory in the goal space into consideration. We denote the distance moved credited to each action a_t as δ_t. Hence, the route length is $\Gamma = \sum_{t=0}^{T-1} \delta_t$. Then, pick the waypoints with interval of $\frac{\Gamma}{k}$ route length.

In this paper, we experimented with both of them, whose results are presented in Sec 5.4. It turns out even the simplest time-sample strategy is an effective way to extract information of goal structure, suggesting much promising results for more heuristic strategies.

Algorithm 1 Hindsight Planner

```
Given:
    • an off-policy RL algorithm $\mathbb{A}$
    • a strategy $\mathbb{S}_{her}$ for sampling goals for replay
    • $k$ and a strategy $\mathbb{S}_p$ for picking $k$ waypoints for replay
1:  Initialize $\pi, \eta$
2:  $B_{rl}, B_p \leftarrow \varnothing$
3:  for episode = $1, M$ do
4:      Sample a goal $g_u$ and an initial state $s_0$.
5:      Planning subgoals $(g_1, g_2, ..., g_k) \leftarrow \eta(\psi(s_0), g), g_{k+1} \leftarrow g_u, j \leftarrow 1$
6:      for $t = 0, T-1$ do
7:          If $g_j$ is achieved, $j \leftarrow min(j+1, k+1)$
8:          Sample an action $a_t$ using the behavior policy from $\mathbb{A}$
9:              $a_t \leftarrow \pi_b(s_t||g_j)$
10:         Execute the action $a_t$ and observe a new state $s_{t+1}$
11:     end for
12:     Pick $k$ waypoints $s_{i_1,...,i_k}$ using $\mathbb{S}_p$, store $(\psi(s_0), \psi(s_{i_1,...,i_k}), g)$ into $B_p$
13:     for $t = 0, T-1$ do
14:         $r_t := r(s_t, a_t, s_{t+1}, g)$
15:         Store the transition $(s_t||g, a_t, r_t, s_{t+1}||g)$ into $B_{rl}$
16:         Sample a set of additional goals for replay $G := \mathbb{S}(currentepisode)$
17:         for $g' \in G$ do
18:             $r' := r(s_t, a_t, s_{t+1}, g')$
19:             Store the transition $(s_t||g', a_t, r', s_{t+1}|g')$ into $B_{rl}$
20:         end for
21:     end for
22:     for $t = 1, N$ do
23:         Sample a minibatch $B_1, B_2$ from the replay buffer $B_{rl}, B_p$, respectively.
24:         Perform one step of optimization over $\pi$ using $\mathbb{A}$ and minibatch $B_1$
25:         Perform one step of optimization over $\eta$ with minibatch $B_2$
26:     end for
27: end for
```

3.2.3 Overall Algorithm. See Algorithm 1 for the overall description of our framework. Note that our critical modifications to the vanilla HER locate at Step 5, 7, 12, 25, in which we fetch out the hindsight waypoints from that episode and train the planner at the replay stage. Actually, our framework is compatible with other improvements of off-policy RL algorithms or HER, such as the prioritized experience replay strategy around Step 24 [11, 23, 33]. Further more, it does not have to cooperate with HER, any RL algorithm applied on decomposable tasks where all subtasks share same goal space is compatible with our framework.

4 RELATED WORK

To learn an agent that would complete multiple tasks, or react to different goal correspondingly, is the inevitable requirement of artificial general intelligence. The goal-oriented policy has drawn numerous efforts to settle down problems in various scenarios, such as learning to grasp objects [17, 26] through imitation learning, and constructing lower-level controller which receives goals sent from higher level policies in hierarchical RL [14, 16]. In the field of learning goal-oriented policy, the reward is extended to condition on goals. The sparse reward is always the natural choice to set up a learning procedure. HER was introduced to deal with the sample inefficiency caused by sparse reward signal. The issue that remains unsettled in HER method has been stated above.

Up to now, there emerge several works to reduce the sample complexity caused by distribution mismatch of goals in HER. Zhao et al. [33] use the prior knowledge of physics systems to prioritize high-energy trajectories in the phase of experience replay, requiring adjustment of energy equations or the weights assigned to multiple kinds of energies for different tasks. Ren et al. [21]generate intermediate task goals from visited states based on the distance between those states and original task goals, working in a curriculum learning way. Note that the distance metric in this method actually works like a kind of dense reward function, which often leads to local optimal. Liu et al. [10] introduce competitive agents to encourage random exploration, lacking informative guidance to realize original goals.

Hierarchical Reinforcement Learning(HRL), in which multiple layers of policies are trained to perform decision-making or control to construct multiple levels of behavioral abstraction, has long held the promise to learn difficult tasks. By having a hierarchy of policies, of which only the lowest one conducts actions to the environment, one is able to train the higher levels to plan over a longer time scale [8]. The difficulties lie in the semantic definition of actions of each policy layer, the joint training without incurring an inordinate amount of experience collection. Previous work has attempted to tackle these difficulties in a variety of ways [3–5, 14, 31]. [7] proposed Hierarchical Actor-Critic(HAC) with hindsight, which also applies the idea of hindsight to train the policy for each level. Our planner is much like the upper-level policy in HRL. However, our method decouples planner and actor more clearly, with the low level only cares about achieving the given goal and the high level only needs to capture the structure of feasible goal space. Free of joint training avoids the inordinate collection of interactions. Besides, the supervised learning paradigm also provides a good property of fast convergence.

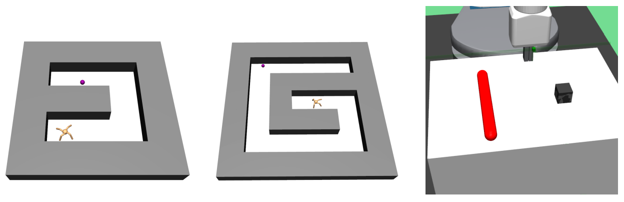

(a) U-maze (b) G-maze (c) Restricted Pushing

Figure 3: Customized Environments

5 EXPERIMENTS

This section is organized as follows. In Sec 5.1 we introduce the environments we test our algorithm on and the experiment setup. In Sec 5.2, we check whether HP can generate legal subgoals for agents. In Sec 5.3 we check whether HP helps to accelerate exploration towards goals of original tasks. In Sec 5.4 we compare the performance of our algorithm with vanilla HER, as well as another baseline proposed by [33]. In Sec 5.5 we analyze the impact of the number of subgoals on the performance.

We upload the video of our experiments on here [1], and the code on here. [2]

5.1 Environments

We use the standard multi-goal environments provided by OpenAI Gym [2, 18]. In addition, we create two ant locomotion environments that have non-linear feasible goal space to test the capability to handle this kind of goal space. We list them out in three categories:

(1) **Fetch Environment**. Including *Reach, Pushing, Sliding, Pick-and-place*. In these tasks, the agent has to control the 7-DOF robotic arm to reach someplace, move or pick one object to the designated spot. For experiments in Sec 5.3, we restrict the *Pushing* by fixed initial state and strip goal distribution, as shown in Figure 3(c).
(2) **Hand-Manipulate Environment**. These tasks require the agent to control a 24-DOF robotic hand to rotate and/or move *Block, Egg* to given orientation and/or position.
(3) **AntMaze Environment**. Including *U-maze, G-maze* which has the maze of shape liking letters 'U' and 'G', respectively. We built them based on the open-sourced Gym Environment and *MuJoCo* [29] physics engine. In these tasks, the 8-DOF quadrupedal ant is initialized at fixed point in the maze, and has to move to goal spot (the purple sphere in Figure 3) sampled randomly at the beginning of each episode.

5.1.1 Experiment Setup. For the hyperparameters used in the actor and critic networks, as well as the training process, we keep consistent with Plappert et al.[18] except that we use 3 MPI workers for *Fetch* tasks and 6 for *Hand* and *Ant* tasks. For planner neural network, we use the hidden size of 64 for LSTM cells, batch size of 128 and adam optimizer with 10^{-3} learning rate for training. Buffer size for planner is 1e4 episodes. The δ_g in Eq. 1 is set to 0.05 for *Fetch* tasks and 0.2 for *AntMaze* tasks. The distance and rotation δ_g for *Hand* tasks are 0.01 and 0.1(rad), respectively. In addition, we loose the δ_g to $2\delta_g$ for subgoals, since rigorous alignment is unnecessary on midway subgoals.

5.2 Does HP generate feasible subgoals?

In order to check whether HP is able to generate legal and feasible subgoals to break down ultimate goals, we visualized subgoals generated in tasks mentioned above using the time-sample strategy of picking waypoints. It is shown in Figure 4 that the subgoals generated by our planner gradually make sense as the learning progresses. HP still works well at planning subgoals for 3d-rotations and complex nonlinear feasible goal space of maze. Note that the planner does not receive any prior knowledge about the goal space (except the normalization of quaternion for getting a valid quaternion).

[1]https://youtu.be/0hL3JRUtyOk
[2]https://github.com/aamas20-984/HindsightPlanner

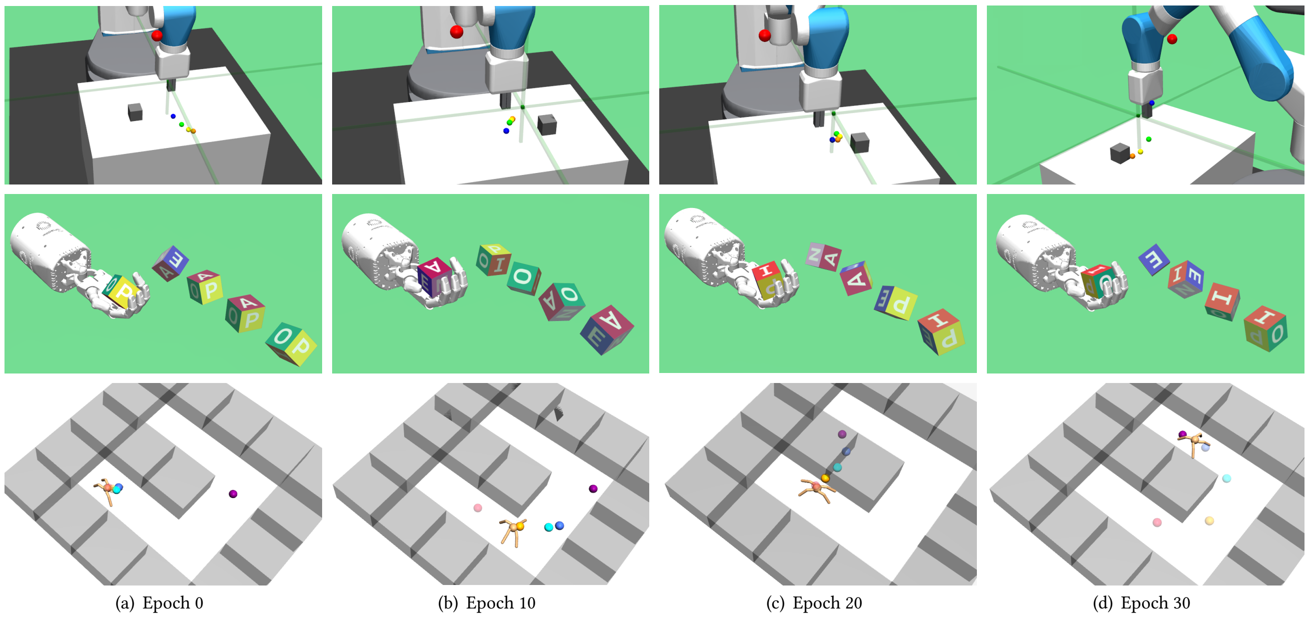


Figure 4: Visualization of subgoals generated by HP on FetchPickAndPlace(top row), HandManipulateRotateXYZ(middle row), AntMazU(bottom row). Subgoals are shown by spheres ordered in brown, yellow, green, blue in Fetch and AntMazeU, and by side blocks arranged from right to left(the right-most one is the initial state and the left-most one is ultimate goal orientation) in HandManipulateRotateXYZ.

For the visibility of generated subgoals, we set the transparency of walls in the maze environment to 20%. The subgoals in Figure 4 are visualized using checkpoints of planner at epoch 0, 10, 20, 30 of the training stage. We could see that subgoals snapshot at epoch 0 are quite meaningless even wrong, while the subgoals snapshot around epoch 30 already become instructive. In our experiments, each epoch consists of 50 episodes, which means the planner could capture the feasible goal space rapidly from the early exploration of the agent and then help the agent to explore towards broader goal space in turn.

5.3 Does HP improve exploration for task with restricted goal distribution?

For the purpose of checking whether our method does help the exploration move towards the goals of original task under the condition that all goals are sampled from an area far away from its initial state, we took the restricted *FetchPush*task for experiment, where the initial states are fixed to one spot and the goals are sampled from line-segment several units away from the initial spot, just as shown in Figure 3(c). This environment setting avoids the effect that uniformly sampled goals could promote the exploration of agents.

We draw the curves of task success rate of our method (using time-sample) and original HER across 4 runs with random seeds in Figure 5. Note that we also plot the averaged success rate curves of these two methods in the unrestricted task, i.e. the curves in Figure 6(b). We can see that our method solved the restricted task at a similar speed as that in the unrestricted version, while HER often

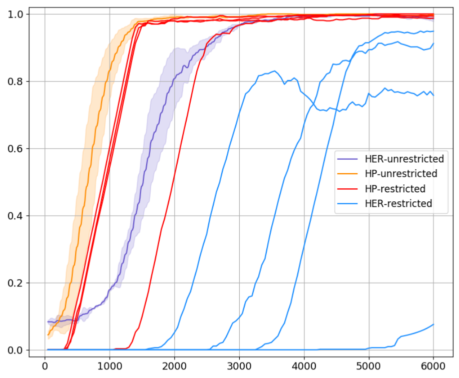
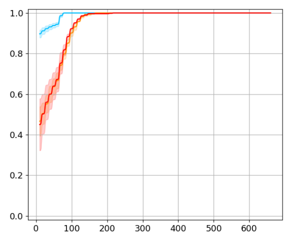


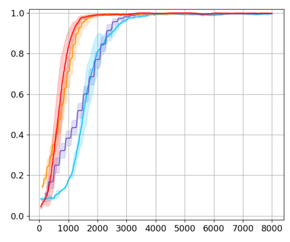
Figure 5: Learning curves in restricted FetchPush task. The y-axis is the success rate and the x-axis is the number of episodes used for training with 3 mpi workers.

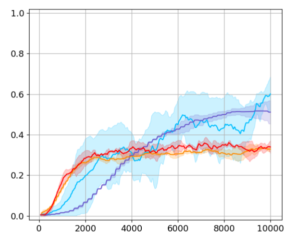
get stuck for a long time without goals sampling uniformly around the initial spot. We can conclude that the subgoals generated by our planner guide the exploration with higher intentionality towards goal distribution of task.

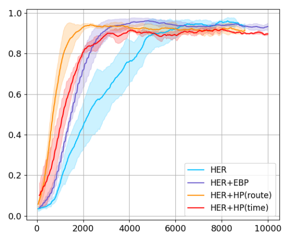
5.4 Does HP improve performance on general tasks?

We then check whether HP improves performance on mentioned tasks against vanilla HER. Moreover, we compared against HER


(a) FetchReach

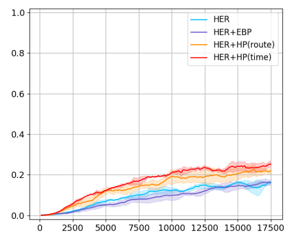

(b) FetchPush


(c) FetchSlide

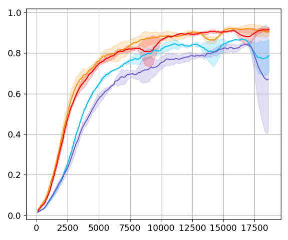


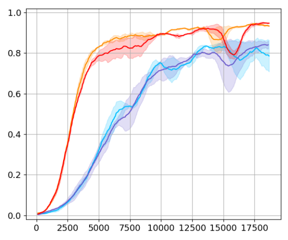
(d) FetchPickAndPlace

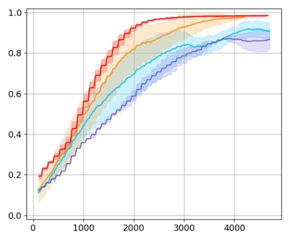
Figure 6: Learning Curves for Fetch environments. The results are averaged across 5 random seeds and shaded areas represent one standard deviation. The red and orange curves correspond to the time-sample and route-sample strategy. The y-axis is success rate and x-axis is the number of episodes used for training with 3 mpi workers.

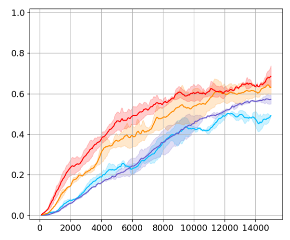


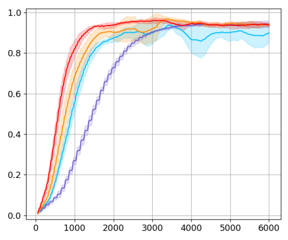
(a) BlockFull

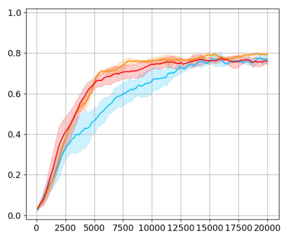

(b) BlockRotateXYZ


(c) BlockRotateParallel


(d) BlockRotateZ


(e) EggFull


(f) EggRotate


(g) AntMazeU

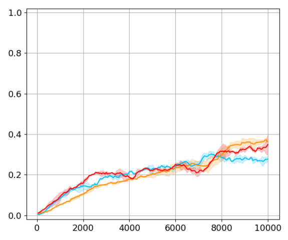
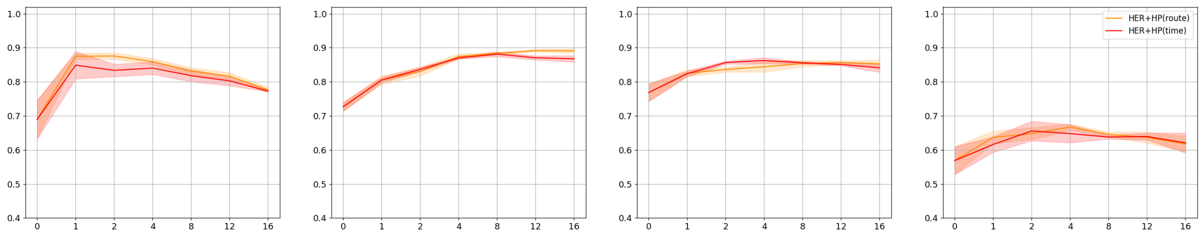

(h) AntMazeG

Figure 7: Learning Curves for Hand and Ant environments. The results are averaged across 5 random seeds and shaded areas represent one standard deviation. The red and orange curves correspond to the time-sample and route-sample strategy, respectively. The y-axis is the success rate and x-axis is the number of episodes used for training with 6 mpi workers.

with Energy-Based Prioritization (HER+EBP)[33]. We tested both of the two strategies proposed in 3.2.2, but without prioritizing trajectories since we want to verify the effectiveness of the strategy with minimal domain knowledge. For both of them, we empirically set the number of subgoals to 4 for all *Fetch* and *Ant* tasks, and 2 for *Hand* tasks. Since *FetchReach* and *AntMaze* do not contain an object to be manipulated, we did not include the HER+EBP versions for them.

From Figure 6 and Figure 7, we can see that the time-sample planner and route-sample planner have minor differences in terms of converged performance, as well as learning speed in some tasks. However, there is one thing need to notice that the route-sampled waypoints may be contorted to segments that have a large span in goal space, which may caused by occasional overaction. Comparing with HER and HER+EBP, the two types of planner exhibit better sample efficiency and higher success rate among most of the 12 tasks. In terms of sample efficiency, our method reduced the number of episodes needed to converge by 50% in *FetchPush*(1500/3000), 60% in *HandManipulateBlockRotateParallel*(5000/12500) against both of HER and HER+EBP. In terms of converged performance, our method lift the success rate by 10% in *HandManipulateBlockFull*, by 20% in *HandManipulateEggFull* against vanilla HER. In *FetchReach*, the task is too easy for HER, while our planner needs some time to accommodate the environment. The only exception is *FetchSlide*, which has to be completed by final strike with proper force in correct direction. HP indeed is not suitable for solving this kind of problem that has to be done at one stroke. The reason why EBP does not improve performance on many rotation tasks might be that the trajectories with higher energy waste too much energy on unnecessary rotations, which instead reduces the agility of agents.



(a) FetchPickAndPlace (b) HandManipulateBlockRotateZ (c) HandManipulateEggRotate (d) AntMazeU

Figure 8: Ablation study of the number of subgoals HP planned for breaking down ultimate goal. (a)-(d) shows the averaged test success rate across all training episodes in four tasks. The x-axis is the number of subgoals, where 0 means no subgoal, i.e. vanilla HER.

5.5 What's the impact of the number of subgoals?

In this section, we experimentally investigated the impact of the number of subgoals. See Figure 8 for the averaged test performance across all training episodes with different number of subgoals in *FetchPickAndPlace, HandManipulateBlockRotateZ, HandManipulateEggRotate, AntMazeU.*

The numbers of training episodes in this section are 9k, 5k, 6k, 20k for PickAndPlace, BlockRotateZ, EggRotate, and AntMazeU, respectively. These numbers are determinated to ensure agents would converged when training ends. 6 mpi workers were used for this section's experiments.

From the plots, we could see that an appropriate number of subgoals leads to the optimal performance, since overmuch subgoals confine the route strictly while the route depicted by subgoals might not be good enough to complete within given timesteps. This is not unexpected since we use the nonprioritized learning objective (Eq.5), therefore the training waypoints sampled from tortuous exploring trajectories let the planner produces winding path. Smarter waypoints sampling strategy may also provide a way to solve this problem, which we intend to explore in future work. Apart from that, we could also see that the greater leap of success rate from $k = 0$ to 1 comparing the successive changes due to the increase of subgoals, which also indicates the great benefit of decomposed goals on the acceleration of learning process. Ideally, comparing to inputting the ultimate goal, inputting a median subgoal can cut down the distance from starting point to the goal by half, thus the promoting effect is most obvious.

6 CONCLUSIONS

For the mismatch between the distribution of hindsight goals and that of original goals in HER, we propose a novel framework called Hindsight Planner to promote the exploration intentionality towards ultimate goal. Our framework exploits collected trajectories to extract the information about feasible goal space, and then do the planning job of decomposing faraway goals into easy-to-achieve subgoals. We formulate this idea as a supervised learning paradigm with selected waypoints as labels of subgoals leading to hindsight achieved goal. For the part of selecting waypoints in the framework, we provide two simple but effective strategies to pick out waypoints from collected trajectories. Extensive experiments demonstrated better sample efficiency and better completion of our framework over vanilla HER and HER+EBP on a collection of robotic tasks. Since our framework does not take extra assumption about the goal space, it provides high generalization ability to many domains. Besides, due to its independence on the trajectory collecting method, it could be combined with most of RL methods, including off-policy and on-policy ones. A future direction would be to enhance the planner power to provide the ability to produce dynamic number of subgoals according to the difficulty of specific tasks.

REFERENCES

[1] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA.* 5048–5058.
[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).
[3] Carlos Florensa, Yan Duan, and Pieter Abbeel. 2017. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012* (2017).
[4] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. 2017. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767* (2017).
[5] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. 2016. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182* (2016).
[6] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.
[7] Andrew Levy, George Dimitri Konidaris, Robert Platt Jr., and Kate Saenko. 2019. Learning Multi-Level Hierarchies with Hindsight. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.*
[8] Andrew Levy, Robert Platt, and Kate Saenko. 2017. Hierarchical actor-critic. *arXiv preprint arXiv:1712.00948* (2017).
[9] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
[10] Hao Liu, Alexander Trott, Richard Socher, and Caiming Xiong. 2019. Competitive experience replay. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.*
[11] Binyamin Manela and Armin Biess. 2019. Bias-Reduced Hindsight Experience Replay with Virtual Goal Prioritization. *arXiv preprint arXiv:1905.05498* (2019).
[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
[14] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information*

Processing Systems. 3303–3313.

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

[16] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2661–2670.

[17] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. 2018. Zero-shot visual imitation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2050–2053.

[18] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. 2018. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464* (2018).

[19] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. 2017. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073* (2017).

[20] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087* (2017).

[21] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. 2019. Exploration via Hindsight Goal Generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. 13464–13474.

[22] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal value function approximators. In *International Conference on Machine Learning*. 1312–1320.

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

[24] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms.

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

[26] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. 2018. Universal planning networks. *arXiv preprint arXiv:1804.00645* (2018).

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[28] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[29] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033.

[30] Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817* (2017).

[31] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 3540–3549.

[32] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

[33] Rui Zhao and Volker Tresp. 2018. Energy-Based Hindsight Experience Prioritization. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*. 113–122.