Journal of Artificial Intelligence Research 77 (2023) 355-376 Submitted 04/2023; published 06/2023

Efficient Multi-Goal Reinforcement Learning via Value Consistency Prioritization

Jiawei Xu XUJW20@MAILS.TSINGHUA.EDU.CN
Shuxing Li LSX20@MAILS.TSINGHUA.EDU.CN
Tsinghua Shenzhen International Graduate School
Shenzhen, Guangdong, China

Rui Yang YANGRUI.THU2015@GMAIL.COM
The Hong Kong University of Science and Technology
HongKong, China

Chun Yuan YUANC@SZ.TSINGHUA.EDU
Tsinghua Shenzhen International Graduate School
Shenzhen, Guangdong, China

Lei Han LXHAN@TENCENT.COM
Tencent Robotics X
Shenzhen, Guangdong, China

Abstract

Goal-conditioned reinforcement learning (RL) with sparse rewards remains a challenging problem in deep RL. Hindsight Experience Replay (HER) has been demonstrated to be an effective solution, where HER replaces desired goals in failed experiences with practically achieved states. Existing approaches mainly focus on either exploration or exploitation to improve the performance of HER. From a joint perspective, exploiting specific past experiences can also implicitly drive exploration. Therefore, we concentrate on prioritizing both original and relabeled samples for efficient goal-conditioned RL. To achieve this, we propose a novel value consistency prioritization (VCP) method, where the priority of samples is determined by the consistency of ensemble Q-values. This distinguishes the VCP method with most existing prioritization approaches which prioritizes samples based on the uncertainty of ensemble Q-values. Through extensive experiments, we demonstrate that VCP achieves significantly higher sample efficiency than existing algorithms on a range of challenging goal-conditioned manipulation tasks. We also visualize how VCP prioritizes good experiences to enhance policy learning.

1. Introduction

Deep reinforcement learning (DRL) has gained significant achievements in solving decision-making problems such as games (Crespo & Wichert, 2020; Oh et al., 2021; Liu et al., 2022) and robotics control (Nguyen & La, 2019; Zhu et al., 2021; Colas et al., 2022). Despite these successes, the sparse reward problem remains challenging. In goal-conditioned tasks with sparse rewards, the agent receives a positive reward only when it achieves the desired goals. Consequently, agents have little chance to obtain informative feedback for decision making. Many widely used RL algorithms, such as TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017), DQN (Mnih et al., 2015), and DDPG (Lillicrap et al., 2015), etc., often fail

©2023 The Authors. Published by AI Access Foundation under Creative Commons Attribution License CC BY 4.0.

in the sparse-reward settings (Andrychowicz et al., 2017; Plappert et al., 2018; Mohtasib et al., 2021).

An effective solution for sparse-reward goal-conditioned tasks is using hindsight knowledge (Andrychowicz et al., 2017). Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) replaces desired goals of failed trajectories with achieved states to obtain positive rewards for policy learning, and it has been demonstrated remarkably effective in a wide range of goal-conditioned tasks (Plappert et al., 2018). Despite the advantage of HER, it still suffers from the large sample complexity (Plappert et al., 2018). To improve the sample efficiency of HER, several works are proposed to explore unfamiliar goals (Ren et al., 2019; Zhao et al., 2019a; Pitis et al., 2020), exploiting collected samples in the buffer (Zhao & Tresp, 2018; Ghosh et al., 2021; Lin et al., 2020; Yang et al., 2022) or trading off exploration and exploitation(Fang et al., 2019).

Exploration and exploitation are two long-lasting challenges in RL (Coggan, 2004; Lazaridis et al., 2020). As concluded by previous works (Oh et al., 2018; Guo et al., 2019; Zha et al., 2020), exploiting certain past experiences can implicitly drive exploration, which inspires us to build an efficient framework for goal-conditioned RL based on prioritized experience replay (Schaul et al., 2015b). In such a framework, the most crucial part is the criteria measuring the importance of different samples for different stages of learning. A few prior works introduce criteria such as trajectory energy (Zhao & Tresp, 2018), entropy (Zhao et al., 2019b), goal diversity (Yang et al., 2021; Dai et al., 2021), and goal similarity (Fang et al., 2019) to replace uniform sampling used in HER. However, most of them make some assumptions about the environments, or they are only suitable for specific types of environments.

In this paper, we propose a new prioritization algorithm, **V**alue **C**onsistency **P**rioritization (VCP), to encourage exploiting past good experiences with or without relabeled goals. Specifically, we train an ensemble of neural networks to learn the Q values and measure the value consistency of each transition by calculating the variance of Q values outputted by different networks. Unlike most previous works that prioritize experiences with higher uncertainty (Janz et al., 2019; Lee et al., 2021; Liang et al., 2022), we sample relabeled experiences with higher value consistency more aggressively. Generally, higher value consistency can be understood as lower value uncertainty. We also show that in sparse-reward tasks, value consistency is positively correlated with the value of samples, which contributes to efficient exploitation and implicitly drives exploration. In our experiments, we demonstrate that assigning higher priority to the relabeled data with higher value consistency can significantly improve sample efficiency than existing baselines and other prioritization-based approaches. Furthermore, we conduct a thorough analysis and visualize how VCP prioritizes good experiences to benefit policy learning.

2. Previous Work

In this section, we review some related works which use prioritized sampling for HER or are based on ensemble neural networks.

Prioritized Sampling for HER. Our method is related to many HER-based prioritization approaches with heuristic measurements. Energy-based HER (HEREBP) (Zhao & Tresp, 2018) evaluates the priority of trajectories according to the kinetic, potential and ro-

tational energies in physics. Instead, Maximum Entropy-based Prioritization (MEP) (Zhao et al., 2019b) assigns the priority according to the entropy of a trajectory. Curriculum-guided HER (CHER) (Fang et al., 2019) defines the proximity score between the achieved goal and the desired goal, and diversity score among selected achieved goals to balance exploration and exploitation. DTGSH (Dai et al., 2021) uses the determinantal point processes to model the goal diversity and calculates the transition priority according to the diversity. Relay-HER (RHER) (Luo et al., 2022) decomposes the original long-horizon task into new sub-tasks and proposes random-mixed exploration strategy to choose achieved goals in typical robot manipulation tasks. A disadvantage of these methods is that defining the criteria requires specific domain knowledge.

Ensemble Approaches. Ensemble neural networks are often used to estimate the uncertainty of data in deep reinforcement learning (Osband et al., 2016). Essentially, the ensemble estimates the posterior distribution of the learning target, predicting similarly on areas with rich data and diversely on areas with insufficient data. Many recent RL algorithms apply ensemble models to enhance exploration in online RL (Osband et al., 2016) or enforce pessimism for offline RL (Bai et al., 2022). Bootstrapped DQN (Osband et al., 2016, 2019) uses an ensemble of random value functions to encourage deep exploration. SUNRISE (Lee et al., 2021) leverages ensemble neural networks to calculate the upper-confidence bound (UCB) for action selection, and re-weight Bellman backups when updating Q-values. Besides, in model-based RL, a series of works (Chua et al., 2018; Janner et al., 2019) adopt ensemble dynamic models for robust dynamics modeling. Considering goal-conditioned RL, the most relevant approach is VDS (Zhang et al., 2020), which uses ensemble Q networks to estimate the value disagreement of goals and samples goals with higher value disagreement for exploration. Our work is orthogonal to VDS because VDS assigns the desired goal in the data collecting phase while we focus on the policy training phase.

Other Work. Some recent works develop curriculum learning for efficient learning of the agent. VACL (Chen et al., 2021) designs curriculum which helps the agent efficiently switch learning tasks and prevents local optimization over single task. CGI (Feng et al., 2022) constructs curriculum which prefers to sample goals with high diversity or expected return to guide the agent to learn from sub-optimal trajectories. Another line of recent works design intrinsic reward function to benefit learning. In this work (Trott et al., 2019), they introduce an auxiliary distance-based reward to encourage exploration and prevents learning from falling into local optima. DRIL (Brantley et al., 2020) use the disagreement among ensemble policies to predict a cost. The agent prefers to sample the state covered by the expert. These works propose curriculum learning or reward-based techniques to improve policy learning, but not applied to HER.

3. Preliminaries

In this section, we introduce preliminaries of goal-conditioned RL and DDPG (Lillicrap et al., 2015) algorithm which is used as a baseline RL method in HER (Andrychowicz et al., 2017) and other compared baselines.

3.1 Goal-Conditioned RL

Reinforcement learning can be described as a Markov Decision Process (MDP) with a tuple (S, A, R, P, γ). At time step t, the agent executes an action a_t conditioned on the state s_t. After the environment receives a_t, the state transits to s_{t+1} and returns a reward r_t. The agent learns a policy $\pi(a|s)$ to maximize the expected discounted return:

$$\pi = \arg\max_{\pi} \mathbb{E}_{\tau,\pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right],$$

where τ is a trajectory $(s_0, a_0, r_0, ..., s_T, a_T, r_T)$ and γ is the discount factor.

In goal-conditioned RL, a desired goal g is given by the environment at the beginning of each episode. Agent's policy is conditioned on a state-goal pair, such that $a_t \sim \pi(\cdot|s_t, g)$, to reach the desired goal. Generally, in the goal-conditioned task with sparse rewards, an agent receives a reward of 0 only when it achieves the desired goal, otherwise it receives a reward of -1:

$$r(s, a, g) = [\psi(s) == g] - 1 \tag{1}$$

where the $\psi(s)$ is the achieved goal in state s. Due to the sparsity of informative feedback, it is challenging for the agent to learn from scratch in such settings.

3.2 Deep Deterministic Policy Gradient and HER

DDPG (Lillicrap et al., 2015) is a widely adopted off-policy RL algorithm for continuous control problems. DDPG learns a Q-function and a deterministic policy parameterized by ϕ and θ respectively. In DDPG, the Q-function is updated by minimizing the following objective:

$$\mathcal{L}^{\rfloor}(\phi) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{B}} \left[\left(Q_\phi(s, a) - \left(r + \gamma Q_{\bar{\phi}}(s', \mu_{\bar{\theta}}(s')) \right) \right)^2 \right], \tag{2}$$

where $\mathcal{B}$ is a replay buffer storing collected samples. Target networks are used to stabilize training. For instance, $Q_{\bar{\phi}}$ is the target Q-function and $\mu_{\bar{\theta}}$ is the target policy. The policy is optimized by maximizing the following objective:

$$\mathcal{L}^{\dashv}(\theta) = \mathop{\mathbb{E}}_{s\sim\mathcal{B}} \left[Q_\phi(s, \mu_\theta(s)) \right]. \tag{3}$$

We follow (Andrychowicz et al., 2017) to implement HER with the universal value function approximation (UVFA) (Schaul et al., 2015a). The Q function and policy with desired goal g are formulated as $Q_\phi(s, a, g)$ and $\mu_\theta(s, g)$, respectively. HER relabels one transition $(s_t, a_t, s_{t+1}, r_t, g)$ in a trajectory τ with achieved goals $g' = \psi(s_{t+n}), 1 \leq n \leq T-t$ and the reward $r'_t = r(s_t, a_t, g')$ is calculated according to Eq (1). This relabeling method is named 'future' relabeling and is generally adopted by numerous prior works (Andrychowicz et al., 2017; Fang et al., 2019; Ghosh et al., 2021; Zhao et al., 2019b; Yang et al., 2022).

4. Methodology

In this section, we introduce Value Consistency Prioritization (VCP), based on an ensemble version of HER+DDPG with multiple Q function heads and actor heads. These ensemble Q networks are used to estimate the value consistency for prioritized sampling.

4.1 The Ensemble Architecture and the Data Flow

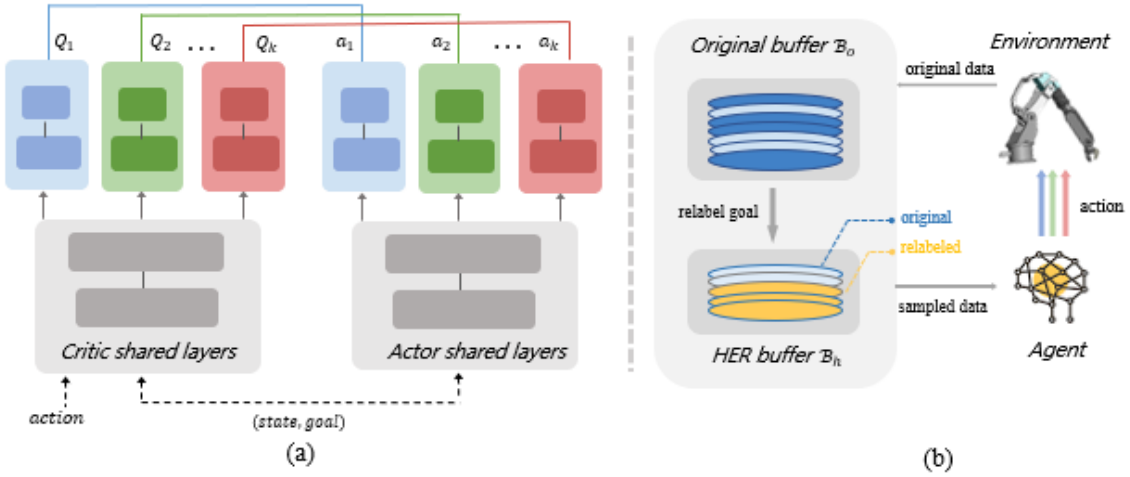

Figure 1: (a) The ensemble architecture in our VCP. (b) The data flow in our learning framework.

The network structure is depicted in Figure 1(a), where both the actor network and critic network are implemented following the multi-head structure. All critics' heads share an underlying embedding network, and all actors' heads share another underlying network, respectively. Each actor head is updated according to its corresponding critic head, and the data used for training each head are sampled independently.

For data collection, all the trajectories collected by different actors are stored in the same buffer, denoted as the original buffer $\mathcal{B}_o$ in Figure 1(b). Then, we uniformly sample a subset of transitions in the original buffer and relabel them randomly following the 'future' method in HER (Andrychowicz et al., 2017). In HER, the sampled data batch consists of a mixture of relabeled and original transitions, where the proportion of the original data is 0.2 by default. In our implementation, we create another buffer to store the mixture of relabeled and selected original transitions and the proportion of original transitions is also set as 0.2. This buffer is referred to as the HER buffer $\mathcal{B}_h$. In addition to the tuple (s, a, g), each data point stored in the HER buffer additionally stores a priority score, which is estimated by the variance of multiple Q-values. Note that we don't need to save the reward r since it can be directly calculated by the reward function in Eq (1). The data flow in our learning framework is illustrated in Figure 1(b).

4.2 Value Consistency Prioritization

Several previous RL algorithms (Janz et al., 2019; Lee et al., 2021; Liang et al., 2022) have concluded that lower Q-value consistency or higher Q-value uncertainty could benefit exploration. Differently, we investigate exploiting samples with higher Q-value consistency for efficient policy learning. Specifically, for the data in the HER buffer $\mathcal{B}_o$, we formally

define the value consistency as:

$$p_i = \left(\frac{\sigma_{max}^2 - \sigma_i^2 + \epsilon}{\sum_{\sigma \in \mathcal{B}_h} (\sigma_{max}^2 - \sigma^2 + \epsilon)} \right)^{\mathcal{T}}, \tag{4}$$

where σ_i^2 is the variance of the estimated Q-values of a transition i, and σ_{max}^2 is the largest variance in the current HER buffer; ϵ is a small positive scalar for numerical stability, and we set it to 10^{-6} in our experiments; $\mathcal{T}$ is a temperature coefficient used to adjust the sharpness of the priority distribution. We set the value consistency p_i as the priority of the transition and normalize the priority as the sampling probability. Intuitively, transitions with larger value consistency would contain smaller Q-value variance.

In the training period, the transitions in the HER buffer are sampled according to the normalized priority p_i to update the policy. Algorithm 1 illustrates how we update the HER buffer with the latest value consistency during training.

4.3 The Overall Algorithm

The proposed algorithm is referred to as **V**alue **C**onsistency **P**rioritization (VCP). We provide the overall algorithm of VCP in Algorithm 2. In our implementation, each head in ensemble DDPG is updated according to its own target network. Since DDPG is an off-policy algorithm, we don't need to distinguish that from which head the data samples in the HER buffer are generated. Therefore, all the heads share the same HER buffer and are updated with different data batches. For each epoch, we also update the priority of each transition in the HER buffer using Algorithm 1.

For data collection, we use all the actor heads to generate multiple trajectories and store them in the original buffer. When updating the HER buffer by sampling data from the original buffer, we use the 'future' relabeling method in HER, and set the proportion of keeping the original data as 20%. In the training period, when we sample a mini-batch from the HER buffer, we normalize the priority by $w_i = p_i / \sum_{\mathcal{B}_h} p_i$ and re-weight the loss with w_i to make the estimation unbiased.

Algorithm 1 Updating The HER Buffer

1: **Input:** the original buffer $\mathcal{B}_o$ and the HER buffer $\mathcal{B}_h$;
2: **while** training is not ending **do**
3: Sample a mini-batch of transitions from $\mathcal{B}_o$;
4: Replace the desired goals with some achieved goals following the 'future' strategy in HER;
5: Calculate the variance of Q-values for each transition in the mini-batch;
6: Store the mini-batch in $\mathcal{B}_h$;
7: Update the priority of each transition in $\mathcal{B}_h$ according to Eq. (4);
8: **end while**

Algorithm 2 The VCP Algorithm

Input: an off-policy RL algorithm $\mathbb{A}$, an original buffer $\mathcal{B}_o$ and a HER buffer $\mathcal{B}_h$;
for epoch $= 0, 1, 2, \cdots, M-1$ **do**
  // **Data generation**
  for head $k = 0, 1, \cdots, K-1$ **do**
    Sample an initial goal g as desired goal and initial state s_o
    for timestep $t = 0, 1, \cdots, T-1$ **do**
      Task an action $a_t \sim \pi_k(\cdot|s_t, g)$
      Observe a new state s_{t+1} and reward r_t
      Store the transition (s_t, a_t, r_t, g) in $\mathcal{B}_o$
    end for
  end for
  // **Training**
  Parallelly run Algorithm 1 in another thread;
  for head $k = 0, 1, \cdots, K-1$ **do**
    Sample a mini-batch of transitions according to the normalized priority in $\mathcal{B}_h$
    Update the k-th head using $\mathbb{A}$ with the mini-batch
  end for
end for

5. Experiment

In this section, we conduct comprehensive control tasks to evaluate the performance of our VCP algorithm and analyze how VCP benefits policy learning. First, we briefly introduce the experiment settings, and the detailed description can be found in the Appendix A-B. Next, we present the comparison results between our VCP algorithm and the baselines, and explain how our algorithm improves exploitation through visual experiments. Finally, we design multiple ablation experiments to demonstrate the effect of each component of our algorithm, and in the Section 5.8-5.9 we analyze the effect of ensemble size on our algorithm.

5.1 Experiment Settings

We evaluate the VCP algorithm on 16 challenging goal-conditioned manipulation tasks, and compare VCP with a set of baselines and other prioritization-based algorithms. Details about the environments are listed in Appendix A. Most of the common hyper-parameters such as learning rate, optimizer, updating rate of target networks are set the same as HER (Andrychowicz et al., 2017). In the stage of data generation, the VCP uses 16 heads and each head collects one trajectory for its own usage. In order to ensure fairness and train all algorithms with the same amount of data, we train other baselines using vectorized environments to run 16 environments in parallel. In this way, the baselines can collect 16 trajectories each time like VCP. After each training epoch, we evaluate each algorithm for 10 episodes, and calculate the average success rate of the 10 episodes. To evaluate VCP algorithm, we use each head to generate 10 trajectories, and compute the average success rate for 160 (10×16) random repetitions. For more detailed settings about experiment,

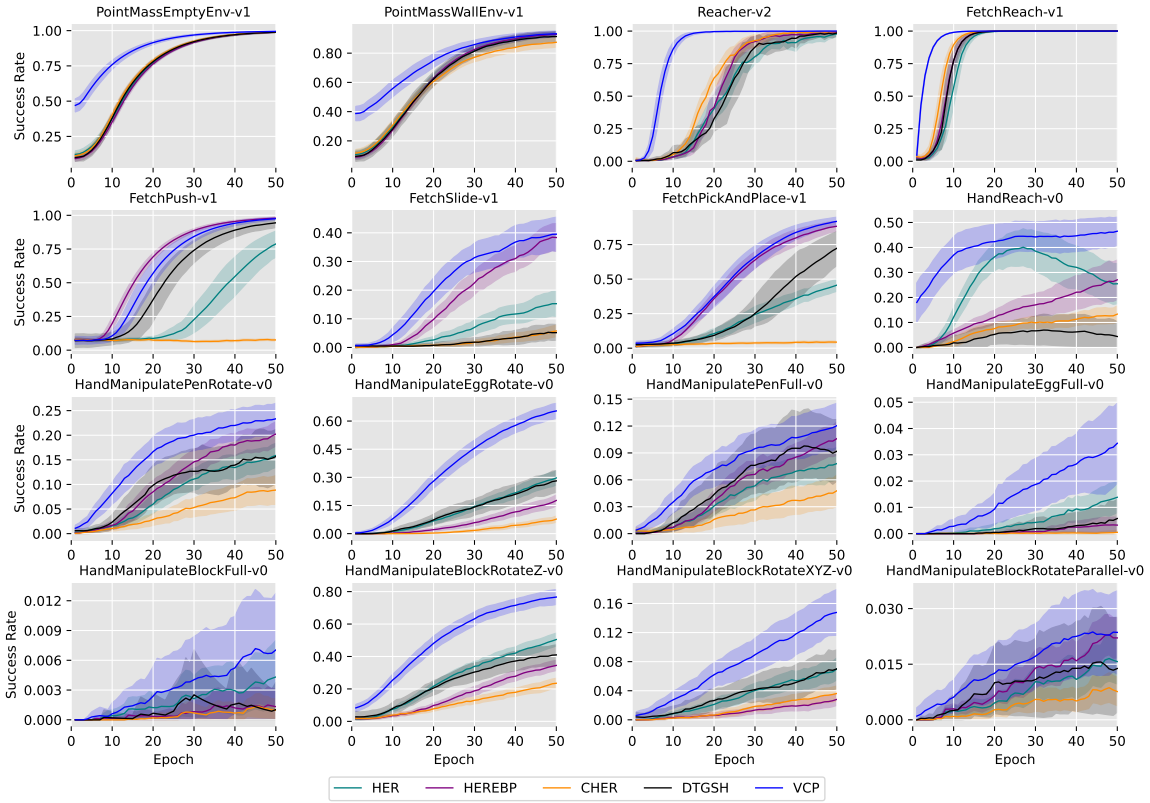

Figure 2: Average success rate (line) with 95% confidence interval range (shaded area) on all environments across 10 random repetitions.

please refer to Appendix B. Our implementation is available at `https://github.com/jiawei415/VCP`.

We compare our algorithm VCP with the following state-of-the-art HER based methods:

- HER (Andrychowicz et al., 2017), which does not use prioritization and uniformly samples transitions from the relabeled data;
- HEREBP (Zhao & Tresp, 2018), which calculates the energy of transitions as the sample priority;
- CHER (Fang et al., 2019), which calculates the proximity and diversity of transitions as the sample priority;
- DTGSH (Dai et al., 2021), which calculates the diversity of the goal with determinantal point processes as the sample priority.

All the compared algorithms are open-sourced, and implemented based on DDPG. So we can use their original implementations to avoid biased evaluation.

5.2 Benchmark Results

Figure 2 shows the comparison results of all the algorithms on 16 sparse-reward goal-conditioned tasks. For each curve in the figure, we average across 10 random seeds and plot the 95% confidence interval with shaded area. Clearly, our algorithm VCP outperforms all the other baselines in 15 out of 16 goal-conditioned environments. In FetchPush-v1, VCP slightly underperforms HEREBP, while it is still competitive compared to other algorithms. In tasks such as HandReach-v0, HandManipulatePenRotate-v0, HandManipulateEggRotate-v0, and HandManipulateBlockRotateZ-v0, VCP significantly improves the final success rate with the same amount of training data, which validates our intuition to exploit samples with higher value consistency.

5.3 Didactic Example

To illustrate how VCP can improve exploitation in HER by prioritized sampling, we use a simple environment, PointMassEmptyEnv-v1, as a didactic example. In PointMassEmptyEnv-v1, the goal of the agent is to reach the target point starting from the starting position. The target point (i.e. the desired goal) is randomly sampled in the map after reset of the environment. Both the states and the desired goals are the XY coordinates. So the achieved goals are equal to the states. For more details about this environment, please refer to Appendix A.

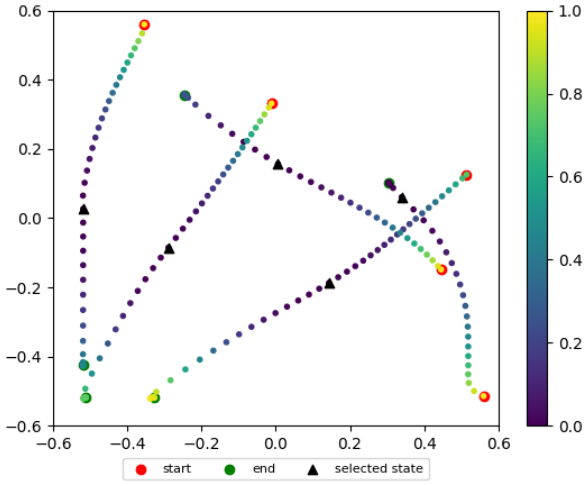

Figure 3: Q-value variance of any state in trajectories.

In Figure 3, we plot five trajectories collected by an agent. The red dot is the starting position of the agent and the green dot is the ending position. We firstly random sampled a state $\hat{s}_t$ (**black** triangle) in each trajectory. Then we calculated the Q-value variance for the combination of this state $\hat{s}_t$ and all other achieved goals (identical to the states) in the trajectory. As shown in Figure 3, the smaller the Q-value variance of the achieved goal, the more likely it is to be selected during the training process. In addition, the achieved goal closer to $\hat{s}_t$ has a higher Q-value, because such an achieved goal is more likely to have a positive reward according to Equation 1. The result in Figure 3 implies that, state-goal pairs with a smaller distance between them exhibit a smaller variance in Q-values, where the distance is generally inversely related to the expected value of the state-goal pair. In VCP, a smaller variance of the Q-value increases the likelihood of selection during the training process. Consequently, our algorithm is more inclined to select data with higher values from the HER buffer to train the model. In this way, our algorithm can increase the chances of training the agent with high-quality data that has a greater expected value, thereby alleviating the sparse-reward problem and improving the exploitation of the relabeled replay buffer.

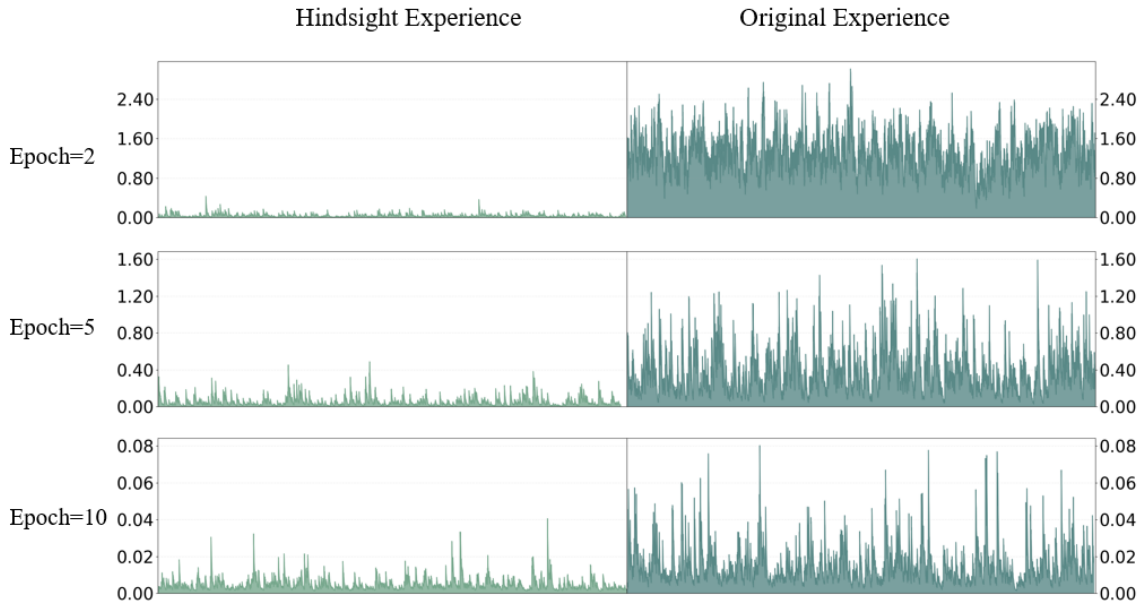

Figure 4: Variance of the Q-values estimated from multiple heads in VCP on the PointMassEmptyEnv-v1 task. The left side denotes the variances of the hindsight transitions (with modified goals) in the HER buffer, while the right side plots the variances of the original transitions. In this illustration experiment, the *ratio* of sampled original data is set to 50%, and another half of data is composed by the hindsight samples. This is to eliminate the impact of data imbalance on evaluating the variances. In our formal experiments, this ratio keeps the value as the same in HER.

5.4 Analysis of Replay Prioritization in VCP

In this section, we continue to analyze the replay prioritization in VCP and explore why sampling data with higher value consistency more aggressively is better for exploitation? We also use PointMassEmptyEnv-v1 task as an example to visualize the variance of the Q-values in training. Figure 4 shows that at different training epochs, the Q-value variances of hindsight transitions are consistently lower than the those of the original transitions. Recalling the advantage of HER on sparse-reward problems, the results in Figure 4 suggests that we should sample transitions with lower Q-value variances because we know that hindsight transitions are generally more informative in the sparse reward setting.

One question here is why the Q-value variances of the hindsight transitions are much lower than those of the original ones. In fact, this is explainable because hindsight trajectories generally achieves the relabeled goals and have higher expected return. On the contrary, the original trajectories often fails to reach the desired goals when learning from scratch. Indeed, when we train the policy for more epochs (i.e., the expected returns of original samples increase), the Q-value variances of the original transitions decrease, as shown in Figure 4. Therefore, we conclude that value consistency is positively correlated with the expected value of samples. If we assign higher priority in transitions with higher value

consistency, we can automatically select more valuable samples and enjoy higher sample efficiency. In this way, our VCP can improve exploitation.

It is worth noting that we are not aiming to replace the original transitions with the relabeled data; instead, we suggest to sample transitions with higher value consistency from the HER buffer $\mathcal{B}_h$, which contains both hindsight experiences and original experiences. The original transitions with higher value consistency (lower Q-value variance) will be sampled with higher probability as well. In this way, each training batch might contain both original transitions and hindsight transitions in VCP, but the ratio between them is not fixed.

5.5 Analysis of Data Reuse Density in VCP

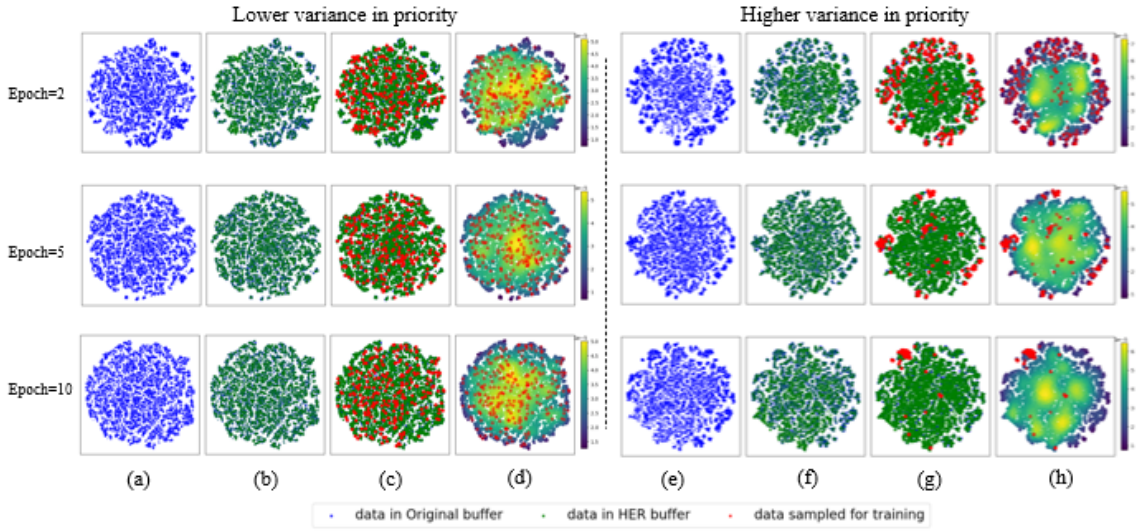

Figure 5: In this illustrative experiment, the *ratio* of sampled original data is set to 50%, and the sizes of the original buffer and HER buffer are set to 10^4. Rows represent different training stages. The first four columns show experiments with lower variance in priority, and the last four columns show experiments with higher variance in priority. The figures are plotted using t-SNE method. The blue points represent the data in the original buffer, the green points represent the data in the HER buffer, and the red points represent the data sampled for training. In (d) and (h), the green data points (HER buffer) are replaced with data density heat map to provide clearer density visualization.

in this section we also visualize the data reuse density to further explain how VCP improve exploitation. We have claimed throughout the paper that VCP can improve exploitation in HER. Now, we provide analysis to support this point with PointMassEmptyEnv-v1 as a toy example. That is, we visualize the data reuse density, which is the estimated Gaussian kernel density of the sampled data, in a 2D space. Since the sampled data is drawn from the hindsight replay buffer, the data points might be re-sampled / reused, we refer to the estimated density as the *data reuse density*.

We use t-SNE (Cieslak et al., 2020) for data visualization. Figure 5 shows the results. Figs. 5(a)-(d) visualize the data reuse density when we sample data points with lower Q-

value variance in priority, which is the sampling strategy in VCP. By contrast, Figs. 5(e)-(h) visualize the data reuse density when we switch the sampling strategy by assigning data points with higher Q-value variance higher priority.

In Figure 5(a), we plot the data distribution of the original buffer, which will be referenced as background in other figures. In Figure 5(b), we plot the data distribution in the HER buffer over the original data distribution. Figure 5(c) further plots the data sampled for training, and Figure 5(d) replaces the data points in HER buffer with data density heat map. As we can clearly observe, in Figure 5(c) and (d) the data points in HER buffer can cover the entire data space, and so do the sampled data points. The results demonstrate that (1) hindsight relabeling itself can provide more fake successful data so that the relabeled data points can cover the entire data space; (2) the sampling strategy focusing on exploiting well fitted hindsight data won't harm the performance of HER. That is, the sampling strategy of lower variance in priority improves exploitation. On the opposite, in Figure 5(g) and (h), the selected (red) data points are concentrated on some regions where the hindsight data points are sparse, and the selected data points fail to cover the data space well. It means that this prioritization replay method cannot exploit the hindsight data well. By aggressively sampling hindsight data with higher Q-value variance, the selected data points are always rarely seen by the agent in the entire training process, and as a consequence the neural networks will never be well fitted. Therefore, we should assign data with lower Q-value variance (i.e., higher value consistency) higher priority.

5.6 Ablation Studies

In this section, we conduct ablation experiments to better understand each components of our algorithm.

Varying Sampling Strategy. We report the performance curves of different sampling strategies, including lower variance in priority (i.e., our method VCP), no prioritization (denoted as VCP w/o prioritization), and the method which samples transitions with higher variance in priority (denoted as VCP w/ higher variance in priority). It is worth noting that the VCP w/o prioritization is equivalent to the original HER with ensemble network. The results are provided in Figure 6. Apparently, VCP significantly outperforms the other two sampling strategies, especially in difficult environments such as the Hand Manipulation tasks. In addition, sampling data with higher variance even deteriorates the performance by a large margin.

Exploration Effect from Multi-Actor. Several previous works show that an ensemble of actors can increase the diversity of sampled trajectories, and this might benefit policy learning as well. Therefore, it is questionable whether the ensemble actors have effect on promoting performance in VCP.

To verify this, we conduct more ablation experiments to see whether VCP benefits from the data diversity introduced by ensemble actors. We design a variant of VCP (denoted as VCP w/o multi-actor diversity) to remove the effect of data diversity from ensemble actors. For this variant, our network structure remains unchanged because we still need multiple heads to calculate the Q-value variance. The difference is that we only use one fixed actor head to interact with the environment at the data generation stage. This setting would exclude the effect of ensemble actors on exploration. Figure 6 illustrates that this variant of

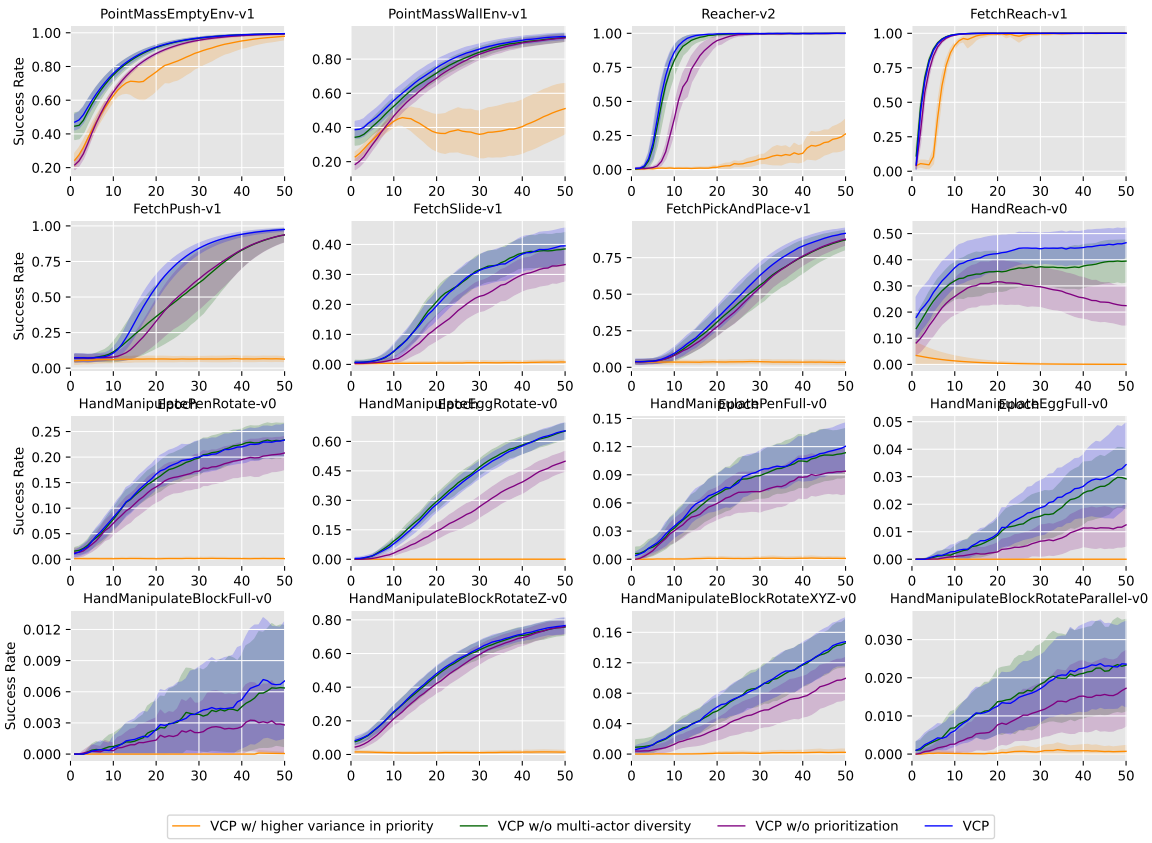

Figure 6: Ablation experiments results on partial environments: average success rate (line) with 95% confidence interval range (shaded area) across 10 random repetitions.

VCP does not show significant performance degradation. In order to eliminate the influence of ensemble networks, we also apply the ensemble structure to baselines for additional comparison. The comparison results in Section 5.9 demonstrate VCP considerably outperforms ensemble CHER.

From the above ablation results, it can be concluded that the prioritized sampling and the prioritization measurement, which is the value consistency, play the most essential roles in our VCP algorithm.

5.7 Training Time

Compared with HER, VCP requires more computation cost due to the usage of ensemble networks. In this section, we compare the training time of VCP with other baselines on the HandReach-v0 task.

Table 1 shows that our VCP algorithm costs a moderate epoch time and is faster than DTGSH. The computation is mainly introduced by updating all the heads in our ensemble

Algorithm	HER	HEREBP	CHER	DTGSH	VCP
Time (min)	1.5	2.74	8.2	34.1	12

Table 1: Average epoch time of VCP and baselines on HandReach-v0 environments.

network. Since VCP achieves considerable improvement over HER, HERBP, CHER and DTGSH, we think the computational cost is worth the performance gain.

5.8 Comparative Experiment on Ensemble Size

We conducted a comparative experiment to analyze the impact of ensemble size on the performance of our algorithm. We selected the best ensemble size, denoted as k, from $[2, 4, 8, 16, 32]$, and the results are shown in Figure 7. As we can see, the VCP with an ensemble size of 16 achieved the best performance. We had already demonstrated through ablation experiments (see Section 5.6) that the diversity data brought by ensemble measure had little impact on our algorithm. Therefore, we explained the impact of ensemble size by visualizing the variance of the Q-values in the HER buffer.

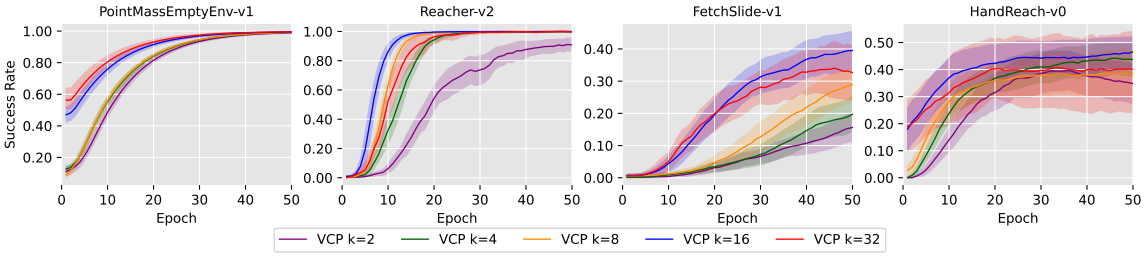

Figure 7: Comparative results on different ensemble size

We continue to use the PointMassEmptyEnv-v1 environment as our toy example, with experimental settings identical to those in Section 5.4. Figure 8 displays the variance of Q-values in the HER buffer under different ensemble size k at the same training epoch (10). The results demonstrate that when the ensemble size is small, the Q-value variances of the hindsight experiences are very low, resulting in the ensemble Q-networks not being able to estimate the value consistency of the hindsight experience effectively. Moreover, this also leads to a very small probability of sampling the original experience, which impairs the learning of the algorithm. This is because the HER (Andrychowicz et al., 2017) has stated that not using the original data will degrade performance.

In addition, we compare the training time of VCP with different ensemble size k in Table 2. It can be observed that despite an exponential increase in ensemble size, the training time does not increase exponentially. This is because we utilize the ensemble network only for the last layer of the network, resulting in higher performance gains with a small parameter cost. Overall, we made a balance between computational complexity and performance, and set the ensemble size k to 16.

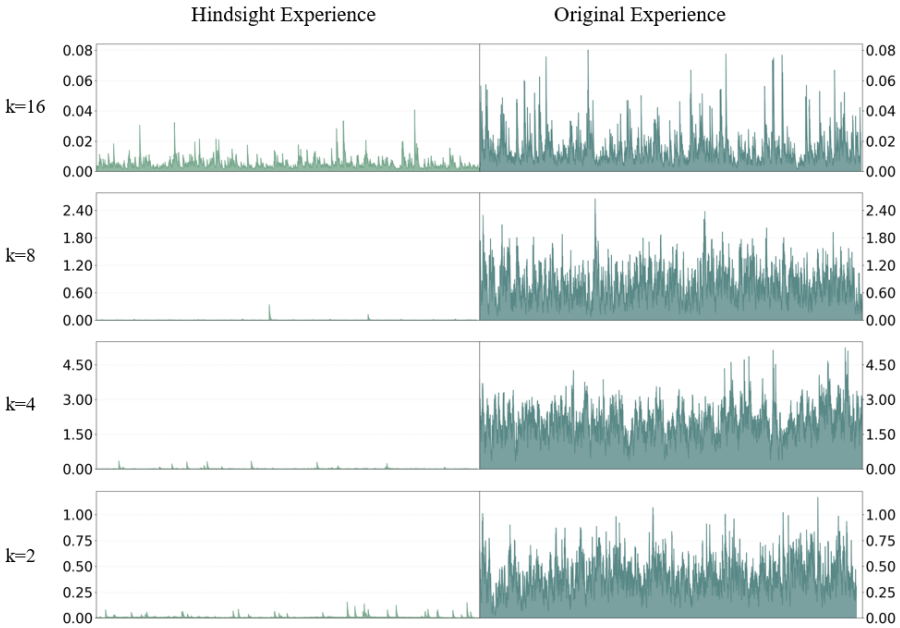

Figure 8: Variance of the Q-values in the HER buffer with different ensemble size. Different rows are the results of VCP under different ensemble numbers k.

Ensemble size k	2	4	8	16	32
Time (min)	3.2	5.6	8.5	12.0	20.7

Table 2: Average epoch time of VCP with different ensemble size on HandReach-v0 environments.

5.9 Comparative Experiment with Ensemble Baselines

Based on Figure 2, our algorithm significantly outperforms the other baselines in 15 out of 16 goal-conditioned tasks. Our VCP has more parameters due to its ensemble network. To eliminate the impact of the number of model parameters, we extended the network of the baselines to the same ensemble structure as ours. In the baselines with an ensemble network, they also collect data using different actor heads and update their multi-head critics like VCP using different batch data. It's worth noting that HER-ensemble k=16 is equivalent to VCP w/ prioritization in Section 5.6.

Figure 9 shows the comparison results of our algorithm with ensemble baselines, where our algorithm outperforms the other baselines in 14 out of 16 environments. Particularly, in the complex Hand Manipulate environment, our VCP algorithm demonstrates significant advantages. Additionally, we compare the training time of our VCP algorithm with that of the ensemble baselines, as shown in Table 3. The training time of our VCP algorithm is competitive with the baselines, and with similar training time, our VCP algorithm achieves better results in most environments.

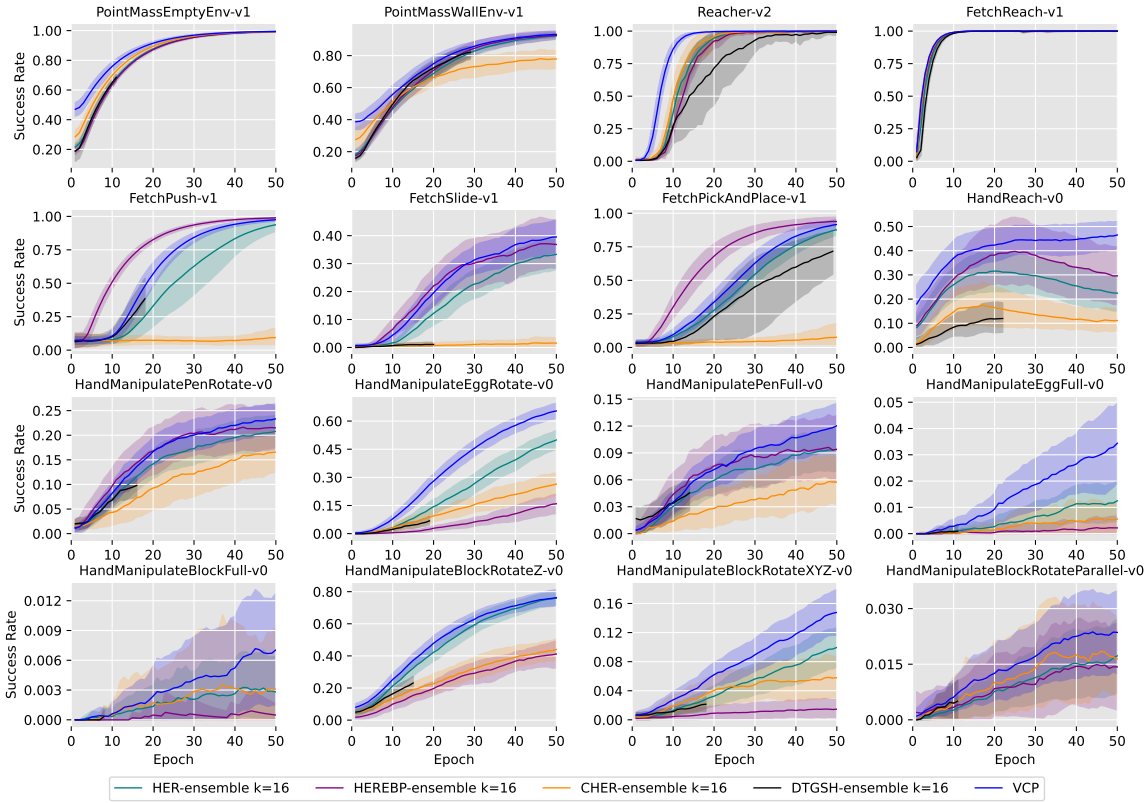

Figure 9: Comparative results with ensemble baselines

Algorithm	HER	HEREBP	CHER	DTGSH	VCP
Time(min)	9.3	11.0	60.5	253.0	12.0

Table 3: Average epoch time of VCP and baselines with ensemble network on HandReach-v0 environment.

6. Conclusion

In this paper, we propose a new value consistency prioritization (VCP) method to improve the exploitation of goal-conditioned RL with sparse rewards. We use an ensemble HER+DDPG structure to estimate the value consistency for different samples, and assign higher priority to the data with higher value consistency. Experimental results show that our algorithm achieves significantly higher sample efficiency than existing goal-conditioned algorithms, and we also explicitly explain how our algorithm improves exploitation through detailed visualization. In addition, our algorithm does not have any assumptions or requirements on the environment and can be easily extended to any goal-conditioned environments. We are interested in applying our method to real robots.

Acknowledgments

The first two authors Jiawei Xu and Shuxing Li contributed equally to this work. This research was supported by the Tencent Robotics X (Rhino-Bird Focused Research Program). The authors wish thank Baoxiang Wang and other anonymous reviewers for their comments.

Appendix A. Environments

We evaluate our algorithm on 16 goal-conditioned manipulation environments, including 9 Hand environments, 4 Fetch environments, 2 Point environments and Reacher-v2. All environments are described here.

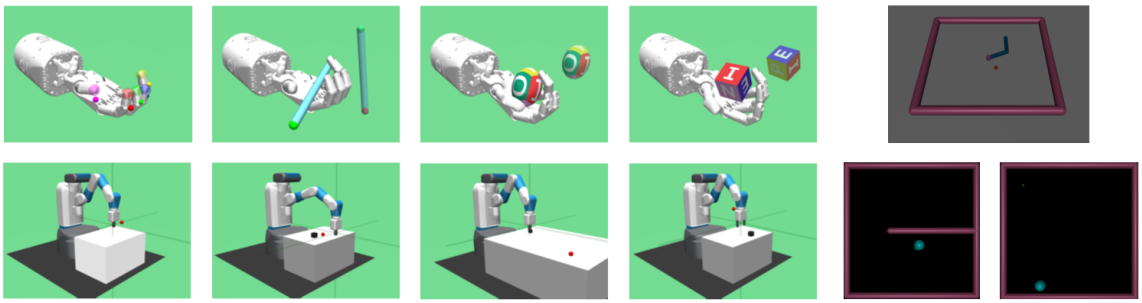

Figure 10: Environment

Hand. These have ethropomorphic robotic hand with 24 degrees of freedom. Of those 24 joints, 20 can be controlled independently whereas the remaining ones are coupled joints. Rewards of these environments are sparse and binary: The agent obtains a reward of 0 if the goal has been achieved and -1 otherwise. Actions are 20-dimensional which are absolute position control for all non-coupled joints of the hand. Observations include the 24 positions and velocities of the robot's joints. There are 9 environments: *HandReach-v0*, *HandManipulatePenFull-v0*, *HandManipulatePenRotate-v0*, *HandManipulateEggFull-v0*, *HandManipulateEggRotate-v0*, *HandManipulateBlockFull-v0*, *HandManipulateBlockRotateParallel-v0*, *HandManipulateBlockRotateXYZ-v0* and *HandManipulateBlockRotateZ-v0*. For a detailed introduction to each environment, please refer to (Plappert et al., 2018).

Fetch. The Fetch environments are based on the 7-DoF Fetch robotics arm, 2 which has a two-fingered parallel gripper. The goal is 3-dimensional and describes the desired position of the object (or the end-effector for reaching). Rewards are sparse and binary too: The agent obtains a reward of 0 if the object is at the target location and -1 otherwise. Actions are 4-dimensional vectors including the desired gripper movement in Cartesian coordinates with 3 dimensions and state (opening or closing) of the gripper with the last dimension. There are 4 environments: *FetchReach-v1*, *FetchPush-v1*, *FetchSilce-v1* and *FetchPickAndPlac-v1*. For a detailed introduction to each environment, please refer to (Plappert et al., 2018).

Point. They are point-based environments, in which the bigger circle aims to reach the smaller point. The state and goal of this environment are both 2-dimensional vectors, which respectively represent the position of the bigger circle and the smaller point, and the action

is also a 2-dimensional vectors, which represents the moving distance of the bigger circle each time. There are two environments in total. *PointMassEmptyEnv-v1*: The bigger circle can reach any position in the environment At the same time, its goal can also randomly appear anywhere in the environment except the walls. *PointMassWallEnv-v1*: There is a wall in the middle which neither the bigger circle nor the small point can reach.

Reacher. It has a robotic arm with 2 degrees of freedom whose task is to reach a particular target. *Reacher-v2* wasn't originally a sparse rewards environment. We modified its state and reward function to make it a goal-conditioned environment with sparse rewards. Its state is an 8-dimensional vectors, which contains sine and cosine of the two rotational joint angles, and the position and velocity of the fingertip in the xy direction. Its goal is a 2-dimensional vectors that represents the position of target, and action is a 2-dimensional vectors, representing the movement of two joints.

Appendix B. Training Details

The hyper-parameters of the HER part of our algorithm are set according to the open source code*. The learning rate of actor and critic is 0.001, buffer size is $1e6$, polyak averaging coefficient for target network updating is 0.95, and number of additional goals used for replay is 4 with the 'future' pattern. Each actor and each critic consists of a 3 layers fully connected network with 256 units in each layer. All actors share the first two layers of the network, as do the critics. We set 16 heads in our implementation. The priority temperature coefficient $\mathcal{T}$ in our algorithm, was chosen among $[1, 5, 9, 11, 15]$ for each environment for best performance. We set batch size to 64 in the FetchReach-v1 and Reacher-v2 environments, and set batch size to 256 for the rest of the environments. We trained for 50 epochs per environment and did not use MPI for parallel data generation like (Plappert et al., 2018). In each epoch, we collect data for n times, and each time we generate 16 trajectories with length 50. We set n to 5 for FetchReach-v1, 15 for Reacher-v2, and 50 for the rest of environment. In each loop, (Plappert et al., 2018) used MPI to generate $19 * 2$ trajectories in parallel. Since we have 16 heads to collect 16 trajectories each time, we indeed use a portion of $16/(19 * 2) = 8/19$ data of (Plappert et al., 2018) to train our agent. This might lead to slight inferior performance of HER compared with the original results reported in (Plappert et al., 2018) in some environments, such as HandManipulateEggFull-v0 and HandManipulateBlockFull-v0. However, the setup is fair for all the compared methods in our experiments.

As mentioned in the paper, all baselines are implemented by their open sourced codes. Since the computational complexity of DTGSH is related to batch size, we keep its original setting 64 in all environments. For a fair comparison, we increase DTGSH's update times to ensure that it uses the same amount of data as other algorithms to update the network. In addition, we run 16 parallel environments for each baseline to produce the same amount of transitions in each epoch. These only differ from the original setting in CHER, HEREBP and DTGSH.

*. `https://github.com/openai/baselines`

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., & Zaremba, W. (2017). Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5055–5065.

Bai, C., Wang, L., Yang, Z., Deng, Z.-H., Garg, A., Liu, P., & Wang, Z. (2022). Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*.

Brantley, K., Sun, W., & Henaff, M. (2020). Disagreement-regularized imitation learning. In *International Conference on Learning Representations*.

Chen, J., Zhang, Y., Xu, Y., Ma, H., Yang, H., Song, J., Wang, Y., & Wu, Y. (2021). Variational automatic curriculum learning for sparse-reward cooperative multi-agent problems. *Advances in Neural Information Processing Systems*, *34*, 9681–9693.

Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, *31*.

Cieslak, M. C., Castelfranco, A. M., Roncalli, V., Lenz, P. H., & Hartline, D. K. (2020). t-distributed stochastic neighbor embedding (t-sne): A tool for eco-physiological transcriptomic analysis. *Marine genomics*, *51*, 100723.

Coggan, M. (2004). Exploration and exploitation in reinforcement learning. *Research supervised by Prof. Doina Precup, CRA-W DMP Project at McGill University*.

Colas, C., Karch, T., Sigaud, O., & Oudeyer, P.-Y. (2022). Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, *74*, 1159–1199.

Crespo, J., & Wichert, A. (2020). Reinforcement learning applied to games. *SN Applied Sciences*, *2*(5), 1–16.

Dai, T., Liu, H., Arulkumaran, K., Ren, G., & Bharath, A. A. (2021). Diversity-based trajectory and goal selection with hindsight experience replay. In *Pacific Rim International Conference on Artificial Intelligence*, pp. 32–45. Springer.

Fang, M., Zhou, T., Du, Y., Han, L., & Zhang, Z. (2019). Curriculum-guided hindsight experience replay. In *Advances in Neural Information Processing Systems*.

Feng, X., Jiang, L., Yu, X., Xu, H., Sun, X., Wang, J., Zhan, X., & Chan, W. K. V. (2022). Curriculum goal-conditioned imitation for offline reinforcement learning. *IEEE Transactions on Games*.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., & Levine, S. (2021). Learning to reach goals via iterated supervised learning. *International Conference on Learning Representations*.

Guo, Y., Choi, J., Moczulski, M., Bengio, S., Norouzi, M., & Lee, H. (2019). Self-imitation learning via trajectory-conditioned policy for hard-exploration tasks..

Janner, M., Fu, J., Zhang, M., & Levine, S. (2019). When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, *32*, 12519–12530.

Janz, D., Hron, J., Mazur, P., Hofmann, K., Hernández-Lobato, J. M., & Tschiatschek, S. (2019). Successor uncertainties: exploration and uncertainty in temporal difference learning. *Advances in Neural Information Processing Systems*, *32*.

Lazaridis, A., Fachantidis, A., & Vlahavas, I. (2020). Deep reinforcement learning: A state-of-the-art walkthrough. *Journal of Artificial Intelligence Research*, *69*, 1421–1471.

Lee, K., Laskin, M., Srinivas, A., & Abbeel, P. (2021). Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 6131–6141. PMLR.

Liang, X., Shu, K., Lee, K., & Abbeel, P. (2022). Reward uncertainty for exploration in preference-based reinforcement learning. In *International Conference on Learning Representations*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, Y., Huang, J., Zimmer, M., Guan, Y., Rojas, J., & Weng, P. (2020). Invariant transform experience replay: Data augmentation for deep reinforcement learning. *IEEE Robotics and Automation Letters*, *5*(4), 6615–6622.

Liu, R.-Z., Pang, Z.-J., Meng, Z.-Y., Wang, W., Yu, Y., & Lu, T. (2022). On efficient reinforcement learning for full-length game of starcraft ii. *Journal of Artificial Intelligence Research*, *75*, 213–260.

Luo, Y., Wang, Y., Dong, K., Zhang, Q., Cheng, E., Sun, Z., & Song, B. (2022). Relay hindsight experience replay: Continual reinforcement learning for robot manipulation tasks with sparse rewards. *arXiv preprint arXiv:2208.00843*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Mohtasib, A., Neumann, G., & Cuayáhuitl, H. (2021). A study on dense and sparse (visual) rewards in robot policy learning. In *Annual Conference Towards Autonomous Robotic Systems*, pp. 3–13. Springer.

Nguyen, H., & La, H. (2019). Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pp. 590–595. IEEE.

Oh, I., Rho, S., Moon, S., Son, S., Lee, H., & Chung, J. (2021). Creating pro-level ai for a real-time fighting game using deep reinforcement learning. *IEEE Transactions on Games*.

Oh, J., Guo, Y., Singh, S., & Lee, H. (2018). Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887. PMLR.

Osband, I., Blundell, C., Pritzel, A., & Roy, B. V. (2016). Deep exploration via bootstrapped dqn. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 4033–4041.

Osband, I., Van Roy, B., Russo, D. J., Wen, Z., et al. (2019). Deep exploration via randomized value functions.. *J. Mach. Learn. Res.*, *20*(124), 1–62.

Pitis, S., Chan, H., Zhao, S., Stadie, B., & Ba, J. (2020). Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pp. 7750–7761. PMLR.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*.

Ren, Z., Dong, K., Zhou, Y., Liu, Q., & Peng, J. (2019). Exploration via hindsight goal generation. *Advances in Neural Information Processing Systems*, *32*.

Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015a). Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015b). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Trott, A., Zheng, S., Xiong, C., & Socher, R. (2019). Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems*, *32*.

Yang, R., Lu, Y., Li, W., Sun, H., Fang, M., Du, Y., Li, X., Han, L., & Zhang, C. (2022). Rethinking goal-conditioned supervised learning and its connection to offline rl. *International Conference on Learning Representations*.

Yang, R., Luo, F., & Li, X. (2021). Combining hindsight with goal-enhanced prediction for multi-goal reinforcement learning. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 314–321. IEEE.

Zha, D., Ma, W., Yuan, L., Hu, X., & Liu, J. (2020). Rank the episodes: A simple approach for exploration in procedurally-generated environments. In *International Conference on Learning Representations*.

Zhang, Y., Abbeel, P., & Pinto, L. (2020). Automatic curriculum learning through value disagreement. *arXiv preprint arXiv:2006.09641*.

Zhao, R., Sun, X., & Tresp, V. (2019a). Maximum entropy-regularized multi-goal reinforcement learning. In *International Conference on Machine Learning*, pp. 7553–7562. PMLR.

Zhao, R., Sun, X., & Tresp, V. (2019b). Maximum entropy-regularized multi-goal reinforcement learning. In *International Conference on Machine Learning*, pp. 7553–7562. PMLR.

Zhao, R., & Tresp, V. (2018). Energy-based hindsight experience prioritization. In *Conference on Robot Learning*, pp. 113–122. PMLR.

Zhu, Y., Wang, Z., Chen, C., & Dong, D. (2021). Rule-based reinforcement learning for efficient robot navigation with space reduction. *IEEE/ASME Transactions on Mechatronics*.