Towards faster policy convergence in a multi-goal, sparse rewards Reinforcement Learning setting

Ojas Joshi
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
oyj@andrew.cmu.edu

Ayush Raina
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
araina@andrew.cmu.edu

Abstract

Most real-world Reinforcement Learning applications suffer from sparse rewards, affecting the scalability and practicality of policy optimization algorithms to these real-world problems, in terms of computing resources and time. In this project, we study and improve over the existing techniques of improving convergence rates of policy optimizations in sparse reward, multi-goal settings. We implement our own approach in an incremental manner on DDPG policy optimization algorithm. However, our approach can be easily scaled across any off-policy Reinforcement Learning algorithm.

1 Introduction

1.1 Motivation

Reinforcement Learning techniques usually involve incremental policy learning iterations guided by a reward signal. However, the performance of these policy optimization techniques largely depends on the reward function being used. Sparse reward settings lead to slower convergence of policy optimization with intermittent and infrequent update signals. Use of reward shaping to overcome this issue is not very scalable as it largely suffers from lack of domain specific knowledge as well as limited knowledge of the nature of admissible behaviour of the agent, as discussed by M. Andrychowicz et al.[1]

Reinforcement Learning methods consist of supervised targets coming from a stream of experience of an agent acting in the environment. Using experience replay[5] helps in resolving the temporal correlations between these sequentially collected transitions and makes the optimization process robust. However, when the agent receives a rare reward signal in a sparse reward setting, this transition gets buried in a big heap of the replay memory collected by the agent. As a result, there is a high possibility of such transitions getting thrown out of the memory without being used for update[7].

The effect of these two, in total, leads to unsolvable or very slow convergence of policy optimization, especially in the earlier part of the training, making these off-policy RL methods impractical to use in real-world problem settings.

1.2 Approach

There have been a lot of successful attempts at improving the efficiency of policy optimization techniques in Reinforcement Learning. Most of them are focused on effective use of higher computation power, better exploitation of experience replay[5], improved ways of collecting agents experience[4], utilizing and coordinating multiple RL agents in a distributed setting[3]. Prioritized Experience Replay[7], builds over the idea of Experience Replay[2] by identifying that some transitions are more

useful to the agent. Consequently, prioritizing these 'surprising' transitions over others during sampling from memory helps the network learn the more important states faster. Hindsight Experience Replay[1](HER) introduces the idea of pseudo-goals and implicit reward shaping. Andrychowicz, Marcin, et al. validate their claim by showing effective speedups in policy convergence rates. Horgan, Dan, et al, 2018, introduce an interesting way of distributing the interactions with the environment across parallel computing systems and effectively learning from these interactions in a smart way.

In this project we study and improve over these techniques used for improving the convergence rates of policy optimization in a sparse reward RL setting. We experiment with multiple ideas mentioned by M. Plappert et. al [6] and introduce two new methods, Prioritized Hindsight Experience Replay(PHER) and Distributed Prioritized Hindsight Experience Replay(DPHER) which is an extension to the former. Synchronous to our goal, the idea behind our approach is two-fold:

1. Implicitly improve the reward density and quality by introducing pseudo rewards and intermediate goals while making sure that the most 'useful' transitions are used for policy updates.

2. Parallelly but effectively explore and exploit the environment.

2 Related Work

Deep Deterministic Policy Gradient[4]

Timothy P. Lillicrap et al. come up with a novel way to extend Deterministic Policy Gradient to continuous action spaces by suggesting an off-policy algorithm based on DQN and DPG. Their approach removes the need of action discretization by augmenting the critic network to output action values directly.

Universal Value Function Approximation[8]

Deep reinforcement learning techniques try to learn an approximation function to the actual value function or the Q-value function. Typically, this value function is usually of the form, $f : S, A \rightarrow IR$, assuming a predefined goal state. As a result, the policies learned by these techniques are goal specific and cannot be easily scaled to a different goal setting. Schaul T. et. al.[1], argue that these value function approximations can be extended to represent a more generalized knowledge about the environment, irrespective of the goal. They develop on the idea by suggesting a universal value function which is also a function of the goal state, such as, $f : S, A, G \rightarrow IR$.

Hindsight Experience Replay (HER)[1]

The motivation for a novel type of experience replay comes from the recurrent difficulty of finding a good reward function scalable across any reinforcement learning setting. The main idea behind HER is to make a smarter use of sparser rewards by suggesting a pseudo reward function. The pseudo reward function builds upon the intuition that during each episode, the agent can learn to reach an intermediate state on the way to the actual goal state. As a result, every transition is stored in the replay memory with the original goal state in addition to a transition having an alternate intermediate goal.

Prioritized Experience Replay[7]

Experience replay helps in resolving the issue of correlated supervision data during policy optimization. However, whenever the agent receives a rare reward signal in a sparse reward setting, such transition is buried in a big heap of the replay memory collected by the agent. During random sampling, this transition has no specific priority and hence no guarantee of getting used for the value function update. Tom Schaul et. al.,2015 exploit this fact to come up with an effective way of prioritizing the transitions in the replay memory for policy optimization updates.

Distributed Prioritized Experience Replay[3]

With a motivation of better and effective use of limited computational power for deep reinforcement learning, Dan Horgan et. al. scale up agent experience generation to a multi-agent setting. They propose a novel and computationally efficient way to distribute the work of experience collection/generation amongst a team of multiple agents. By separating the experience collection

part('actor') from the learning agent('leaner'), they accelerate experience generation by multiple-folds.

3 Methods

3.1 Prioritized Hindsight Experience Replay

In a standard multi-goal, sparse-reward setting, learning the goal states is the most critical part during the initial part of training. Using TD-error of the transition as priority, Prioritized Replay makes sure that the agent learns the goal state once it reaches that goal state during exploration. However, this approach still suffers from sparse rewards and is highly dependent on the exploration policy of the agent. On the other hand, the crude idea behind HER is learning the path to the goal in steps, by learning the sub-goals on the way. Consequently, by providing these pseudo-goals, the number of explored goal states increases K-fold, helping the agent to learn faster. However, HER in turn suffers from the problem of important transitions getting buried in the replay memory and never getting sampled for updates. We observe that use of latter can be used to mask the shortcomings of the former or vice-a-versa.

While Hindsight Experience Replay motivates implicit reward shaping, Prioritized Experience Replay(PER) helps with identifying most 'surprising'[7] transitions in the replay and making sure that they are used for the policy updates on higher priority. We propose that using a Prioritized memory along with HER could accelerate the learning further, by sampling the transitions with pseudo-subgoals more frequently, hence reducing the effect of sparse rewards two-folds.

Our agent maintains a prioritized experience replay while adding pseudo reward transitions similar to Hindsight Experience Replay. Following Schaul, Tom, et al, 2015, we save every transition with a corresponding priority value and sample the transitions for policy updates based on the 'proportional' variant of Prioritized Experience Replay. New transitions are weighted with the highest possible priority as they are not associated with a TD-error until they are sampled for an update. As suggested by Schaul, Tom, et al, 2015, we also weigh the gradient updates to the critic network with the corresponding importance weights to anneal the bias introduced by prioritized sampling. After every episode we add the hindsight replay by sampling the pseudo goals using 'future' or 'episode' [1] strategy and use the $'compute\_reward()'$ reward function provided in the OpenAI environments for the pseudo-rewards. For the 'future' strategy, the pseudo goal states are sampled from the states visited after the given state in that episode while in 'episode' strategy any state from the current episode is sampled for hindsight replay. The actor and critic networks used for DDPG implementation are based on Schaul, Tom, et al.,2015 idea of universal function approximators. We use the wrapper utility provided by OpenAI gym to concatenate the goal state with the observation state as the input to the actor and critic networks.

During implementation, as suggested by Andrychowicz, Marcin, et al., 2017, we clip the targets of the critic network to the possible range enforced by the reward function of the environment, $[\frac{-1}{1-\gamma}, 0]$ and also clip the action values generated by the actor network to $[-5.0, 5.0]$. To deal with vanishing gradient problem associated with the $tanh$ activation in the actor network, we add an average of the sum of squared values of the pre-tanh layer with weight=0.01 to the actor loss. While we experiment the effect of different parameters, the fixed parameters have been derived from Schaul, Tom, et al.,2015 and Andrychowicz, Marcin, et al.,2017. For annealing the sampling bias, we normalise the importance weights just before updating the learner unlike Schaul, Tom, et al.,2015. We base our experiments on OpenAI environment FetchReach-v1. However, we give an option of scaling the approach to other robotics environments in the OpenAI Robotics environment suite. Algorithm 1 shows the detailed pseudo-code of our implementation. The code used during experimentation is available at: `https://github.com/ojasjoshi/distributive-prioritised-hindsight-experience-replay`

3.2 Distributed Prioritized Hindsight Experience Replay

While working towards our goal of making policy optimization methods converge faster, we further escalate our approach to use computation power more efficiently and in a smarter way. We follow Horgan, Dan, et al., 2018 and separate the "actor" network, which interacts with the environment, from the "learner" network, which learns from the interactions of the actor. We build over their idea and implement a distributed setting in our approach. We implement two actor networks, each having

Algorithm 1 Prioritized Hindsight Experience Replay with DDPG

Given: Replay Memory size N, mini-batch k, train batch B, hindsight replay parameter K, exponents α, β, Goal Sampling Strategy S_s, pseudo-reward function R_s, Replay Period R_a, Actor network Q, Critic Network μ

Initialize: p_{max} = 1.0, Replay Memory $H = \phi$, Episode Memory $E = \phi$, $\Delta = 0$, actor network parameters θ^{μ}, critic network parameters θ^{Q}

for episode = 1 **to** M **do**
  Sample episode goal g, state S_0 and choose $A_0 \sim \mu(S_0||g)$
  Add exploration noise $A_0 \leftarrow A_0 + N_T$
  for t = 1 **to** T **do**
    Observe $S_t, R_t, is\_terminal$
    Store transition $(S_{t-1}||g, A_{t-1}, R_t, \gamma_t, S_t||g)$ in H with maximal priority p_{max}
    Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in E
    if $t \equiv 0 \mod R_a$ **then**
      for $j = 1$ **to** k **do**
        Sample transition $j \sim P(j) = p_j^{\alpha} / \sum_i p_i^{\alpha}$
        Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
        Compute TD-error* δ_j as,
        TD-error* = $R_j + \gamma_j Q_{\text{target}}(S_j, \arg\max_a Q(S_j, a))$ - $Q(S_{j-1}, A_{j-1})$
        Update transition priority $p_j \leftarrow |\delta_j|$
        Update maximal priority p_{max}
        Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_{\theta^Q} Q(S_{j-1}, A_{j-1})$
      end for
      Update critic weights $\theta^Q \leftarrow \theta^Q + \eta \cdot \Delta$, reset $\Delta = 0$
      Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=S_i, a=\mu(S_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{S_i}$$

    end if
    Choose action $A_t \sim \mu(S_t||g)$
    Add exploration noise $A_t \leftarrow A_t + N_T$
  end for
  for t = 1 **to** T **do**
    Sample goals for Hindsight Replay $G \sim S_s$
    for $g' \in G$ **do**
      Compute reward $R'_t = R_s(S_t)$
      Store transition $(S_{t-1}||g', A_{t-1}, R'_t, \gamma_t, S_t||g')$ in H with maximal priority p_{max}
    end for
  end for
  Reset Episode Memory $E \leftarrow \phi$
end for

|| Denotes Concatenation
$*$ Assumes concatenated goalstate

their own instance of the environment and each actor having its own network. From time to time, we update the actors with the learner parameters. In addition, each actor has it's own experience replay memory and a common prioritized memory accessible by the learner as well. The learner memory is updated by sampling transitions from the actor memories in batches.

We first experiment with unprioritized actor memories which randomly samples transitions from the actor memory for learner memory updates. In our main approach, we prioritize the actor transitions based on the TD-errors associated with their potential updates. In addition, while updating the learner memory with transitions from the prioritized actor memory, we pass the corresponding priority associated with that transition. This saves the double computation of TD errors while also keeping the transitions updated in the learner memory. This approach has an advantage over Prioritized

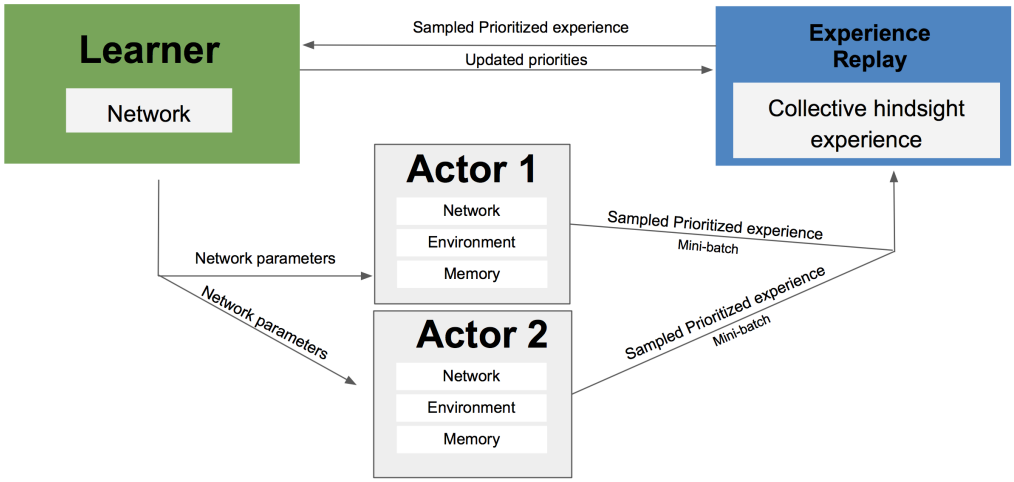

Figure 1: Graphical Model of DPHER

Experience Replay as the incoming transitions are now already prioritized by their TD errors. This reduces the mandatory but nonessential advantage given to new transitions in Prioritized Experience Replay [7]. Prioritized actor memories also keep the bad transitions out of the learner memory by sampling them with lesser priority.

However, prioritized actor memories increase the sampling bias two-fold while also reducing exploration, especially when the target networks are not very well trained and the TD-errors are not representative of the true value in the early part of the training. To overcome the exploration problem, we set one of our actor to have high prioritized sampling with $\alpha = 0.8$ and the second actor to have a more random sampling with $\alpha = 0.3$. We also experiment with increasing the alpha for both the actors as the training progresses, inducing higher exploration in the earlier stages and simultaneously increasing the exploitation. To overcome the issue of high bias induced by priority sampling, we increase the bias annealing in the learner with a higher β factor.

We suspect that the deep correlations between the hyper parameters, actor memory size, batch size of transitions sampled from actor memory into the learner memory and the trade-off between exploration and exploitation induced throughout the training will affect the learning performance to a larger degree.

4 Experiments and Inferences

We test our approach on sparse reward OpenAI robotics environment, FetchReach-v1. The goal of the agent is to learn to move it's end-effector to the goal position as fast as possible. For every step the robotic arm does not reach the goal state, it receives a reward of -1 and a reward of 0 on reaching the goal state. We use accuracy(success rate) metric to evaluate our agents performance. Accuracy refers to the number of times the agent reaches the desired goal after every step, on average. As our motivation stems from faster policy optimization, we study the effect of our algorithms in the first 40 epochs(1 epoch = 50 episodes = 50x50 steps) of training in each case. The plots show the accuracy variations during training time. For the comparative study of our approaches(shown later in Figure 9) we test the trained agent after every epoch for 10 episodes and plot the mean and variance of accuracy over these 10 episodes.

4.1 DDPG and Prioritized Experience Replay(PER)

We first run vanilla-DDPG for 200000 steps of training on the environment. As seen in Figure 1, the agent does not seem to learn at all within 40 epochs, with accuracy never increasing beyond 1%. In this approach, we test if prioritized experience replay helps the agent solve the environment. Figure 1 shows that PER does not seem to help either, with the agent still failing to learn to reach the desired goal state.

4.2 DDPG and Hindsight Experience Replay(HER)

To validate the results shown by M. Andrychowicz et al.[1], we implement hindsight experience replay with DDPG using "future" and "episode" strategies for sampling the pseudo-goals. As seen in Figure 3, DDPG + HER shows tremendous improvement in learning with agent able to learn the environment almost completely within 40 epochs. We test how the number of pseudo-goals(K) sampled during hindsight replay affect the performance in case of both, "future" and "episode" strategies. We see a similar trend as observed by M. Andrychowicz et al.[1], with K=4 performing the best in both cases (See Figure 3) and "episode" strategy outperforming "future" strategy by a small margin.

When the value of K increases, the replay memory fills faster. If the rate at which the replay memory fills is high, the transitions in the memory are refreshed very fast. As a result, the performance of the agent is highly dependent on the chance of a good transition being sampled for learning updates, resulting in poor performance. On the contrary, when K is very small, the number of pseudo-goal transitions is smaller and hence the effect of HER decreases. This makes the choice of K an optimization step with extreme values of K worsening the performance.

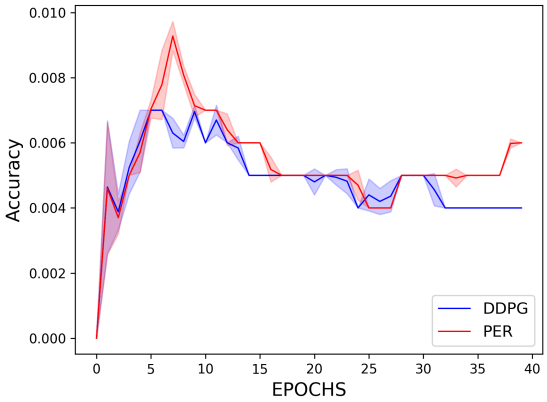

Figure 2: DDPG vs PER

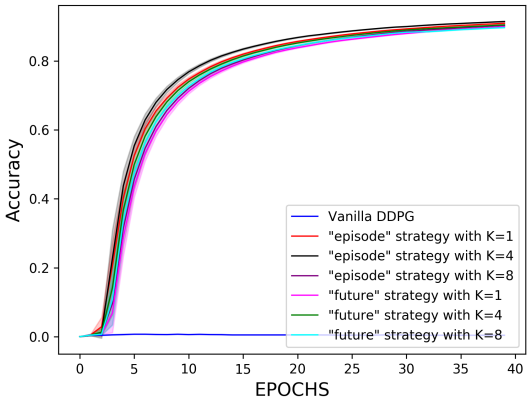

Figure 3: Varying K in HER

4.3 Prioritized Hindsight Experience Replay(PHER)

With expected results obtained in the implementation of DDPG+HER, we analyze our Prioritized Hindsight Experience Replay approach. Figure 4 shows how number of pseudo goals(K) sampled during hindsight replay affects the performance of the agent. Unlike HER, PHER performs the best with K=8 in case of both, "future" and "episode" strategies with "episode" strategy again outperforming the "future" strategy.

In reference to HER, prioritizing the memory increases the chances of sampling the good transitions before they are replaced by the incoming transitions. As a result, this gives a margin to increase the value of K further as compared to HER and the best performing value of K jumps from 4 to 8 accordingly.

We also study how the degree of memory prioritization affects PHER. We test the algorithm with values of alpha=0.3(low prioritization), alpha=0.7, and alpha=0.9(high prioritization) using "future" strategy. Figure 5 shows that alpha=0.3 marginally outperforms alpha=0.7 & 0.9 with alpha=0(no prioritization) performing the worst. Values of alpha close to 1, implicitly decrease the exploration of the agent by biasing only some transitions for learning updates and result in poorer performance especially in the early part of training. More exploration in the earlier part of the training, helps the agent perform better, as seen in Figure 7.

4.4 Distributive Prioritized Hindsight Experience Replay(DPHER)

We test our DPHER approach with 2 actors and one learner. DPHER involves numerous interlinked parameters such as actor memories, actor mini-batch size, prioritization of actors, strategies for

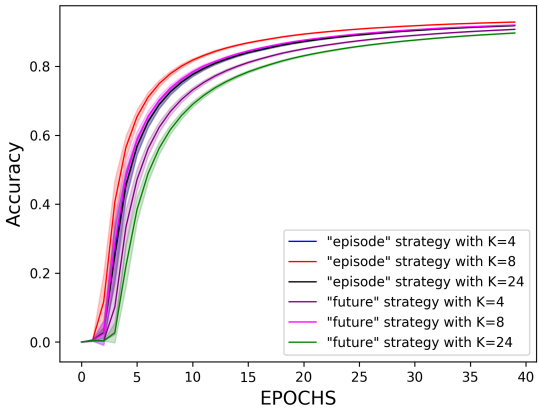

Figure 4: Varying K in PHER

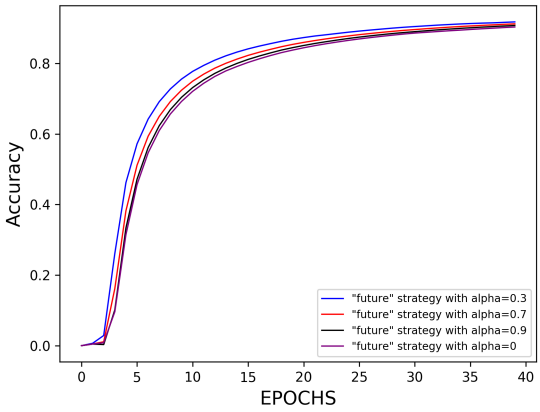

Figure 5: Varying Prioritization in PHER

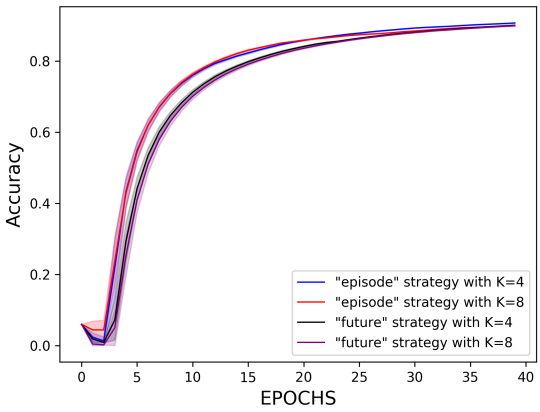

Figure 6: Varying K in DPHER

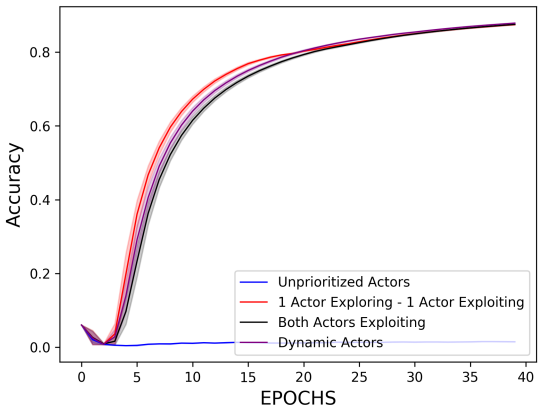

Figure 7: Varying Actor exploration strategies

pseudo-goal sampling etc. which makes optimizing these hyper-parameters exponentially harder. As a result, the trends observed in the following experiments could be domain biased or be environment specific. However, we try to show the most important trends among these correlations and explain why these trends might be valid across in general. Figure 6 shows the results for varying K and HER strategy. It can be seen that the performance is similar for values of K=4 and K=8. However, "episode" strategy performs better than "future" strategy for sampling pseudo-goals which is consistent with previous observations.

The value of alpha denotes the degree of memory prioritization, which as discussed in PHER relates to the degree of exploitation and exploration of the agent. In case of DPHER, highly prioritized actors bias the transitions fed to the learner memory and the learner memory never receives the transitions 'explored' by the agent. To test the effect of this bias we test DPHER for 3 cases:

1. 1 actor with small alpha value (= 0.3)
2. Both actors with large alpha value (= 0.7)
3. Both actors with small alpha value for first 5 epochs, large alpha value afterwards

As seen in Figure 7, Case 1 (Red), which denotes higher agent exploration, performs better during the initial part of training as the agent benefits from agent exploration but then worsens. On the contrary, for Approach 2 (Black), inhibited exploration slows the training initially. Approach 3 (Purple) benefits from both exploration and exploitation and performs well throughout.

Next we experiment with varying values of actor memories and actor mini-batch sizes. These parameters show significant differences in performances as seen in Figure 8 & 9. During these experiments to easy complexity, we fix the learner memory to a size of 100000 and learner batch-size to 128.

Figure 8,9 shows that larger actor memories perform much better in comparison to smaller actor memories as smaller actor memory suffers from faster refreshing transitions. Incoming transitions and most 'surprising' transitions have equal priority in the actor memory. As a result, smaller mini-batch size for sampling transitions from actor memory into learner memory is unable to sample enough number of 'surprising' transitions. With smaller actor memory and smaller batch-size, the performance is the worst as 'useful' transitions get replaced in the actor memory before they get sampled. Using very high actor batch-size as a remedy for extracting most of the important transitions from actor memory makes the learner memory refresh too fast, degrading the performance. Accordingly, Figure 8,9 show that a bigger actor memory and an optimal value of batch-size of 8 performs the best.

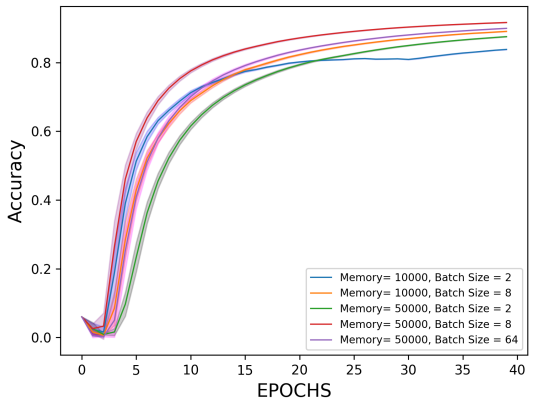

Figure 8: Varying Actor Memory ("Future")

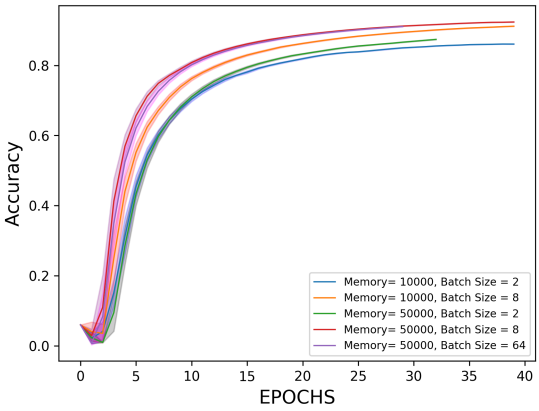

Figure 9: Varying Actor Memory ("Episode")

5 Conclusion

Based on our experimental analysis, we compare the best performing hyper-parameters for our approaches in comparison to the baseline algorithms DDPG+Hindsight Experience Replay(HER), DDPG+Prioritized Experience Replay(PER) and Vanilla-DDPG. Figure 10 shows that Vanilla-DDPG and PER perform very poorly, PHER largely outperforms HER, while DPHER shows a marginal improvement over PHER. These results show that prioritized replay memory coupled with Hindsight Experience Replay significantly helps the policy learning in a sparse reward setting and results in faster convergence. In addition, multiple actors help utilize the computational resources in a more smarter and efficient way while also showing marginal improvements in policy learning.

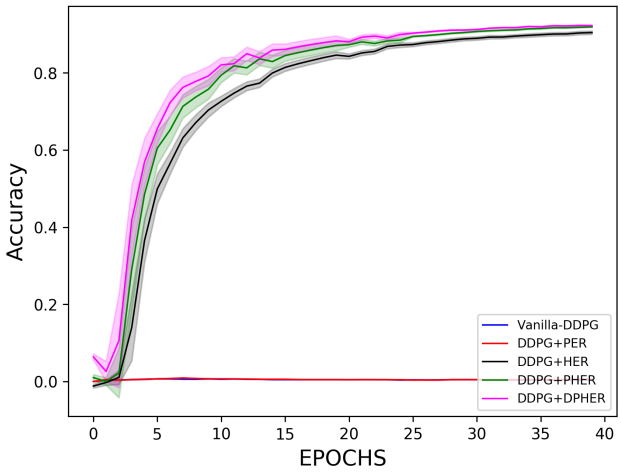

Figure 10: Comparative test performance

References

[1] Marcin Andrychowicz et al. “Hindsight Experience Replay”. In: *CoRR* abs/1707.01495 (2017). arXiv: 1707.01495. URL: http://arxiv.org/abs/1707.01495.

[2] Greg Brockman et al. “OpenAI Gym”. In: *CoRR* abs/1606.01540 (2016). arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540.

[3] Dan Horgan et al. “Distributed Prioritized Experience Replay”. In: *CoRR* abs/1803.00933 (2018). arXiv: 1803.00933. URL: http://arxiv.org/abs/1803.00933.

[4] Timothy P. Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *CoRR* abs/1509.02971 (2015). arXiv: 1509.02971. URL: http://arxiv.org/abs/1509.02971.

[5] Long-Ji Lin. “Self-improving reactive agents based on reinforcement learning, planning and teaching”. In: *Machine Learning* 8.3 (May 1992), pp. 293–321. ISSN: 1573-0565. DOI: 10.1007/BF00992699. URL: https://doi.org/10.1007/BF00992699.

[6] Matthias Plappert et al. “Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research”. In: *CoRR* abs/1802.09464 (2018). arXiv: 1802.09464. URL: http://arxiv.org/abs/1802.09464.

[7] Tom Schaul et al. “Prioritized Experience Replay”. In: *CoRR* abs/1511.05952 (2015). arXiv: 1511.05952. URL: http://arxiv.org/abs/1511.05952.

[8] Tom Schaul et al. “Universal Value Function Approximators”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1312–1320. URL: http://proceedings.mlr.press/v37/schaul15.html.