LEARNING GOAL-CONDITIONED VALUE FUNCTIONS WITHOUT GOAL REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-goal reinforcement learning addresses the tasks where goal specifications are required. State-of-the-art methods in this field~~,~~ utilize goal-conditioned value functions in their operation. Estimating these functions operate on the assumption that the achievement of goal states are tied with large reward. Our first contribution is the redefinition of these functions as expected cumulative path rewards that allows for equally efficient learning in the absence of these goal rewards. This formulation of the goal-less learning of goal-conditioned value functions obviates the requirement of reward-recomputation that is needed by state-of-the-art MGRL algorithms. This futher leads to substantially improved reward sample complexity which is our second contribution. Our third contribution is the extension of the Floyd-Warshall Reinforcement Learning algorithm from tabular domains to deep neural networks.

1 INTRODUCTION

Many tasks in robotics require the specification of a *goal*. For example, a robotic arm may need to move an object to a goal position on a table (Gu et al., 2017). ; A mobile robot may be tasked with navigating to a goal landmark ~~in~~ a map (Zhu et al., 2017). The adaptation of reinforcement learning to such goal-conditioned tasks is called Multi-Goal Reinforcement Learning (MGRL)(Plappert et al., 2018b).

State-of-the-art MGRL algorithms (Andrychowicz et al., 2017; Pong et al., 2018) work by estimating *goal-conditioned value functions* (GCVF). A GCVF is defined as the expected cumulative reward of reaching from a start state conditioned on the given goal state. They are used to compute the actions to be taken at any state, also known as the *policy*.

These algorithms require the use of *goal-rewards* , which is the dependence of reward functions on the desired goal, $r(s, a, g)$. For example, in the Fetch-Push task (Plappert et al., 2018b) of moving a block to a given location on a table, every movement incurs a "-1" reward while reaching the desired goal ~~earns~~ "0" reward. Although~~t~~ this dependence allows for the association of high reward with the reaching of goals, it incurs additional costs. In Hindsight Experience Replay (HER) (Andrychowicz et al., 2017), failed experiences are re-evaluated as succesful ones by assuming traversed states to be desired goals. Due to the above dependence, the association of every pseudo-goal with high reward requires an independent *reward-recomputation* step.

~~The goal-rewards requirement is avoidable.~~ Consider a student who has moved to a new university. To learn about the campus, the student explores it randomly with no specific goal in mind. When tasked with finding a goal classroom, the student can then use the learned connectivity of the campus to make their way to class ~~in~~ the shortest possible explored path. The key intuition here is that the student is not incentivized to explore specific goal locations (i.e. no goal rewards). Based on this intuition of goal-less learning, we redefine GCVFs to be the expected *path-reward* that is learned for all possible start-goal pairs. Instead of goal rewards, we introduce "a"? *one-step loss* that assumes single steps paths to be the maximum reward paths between adjacent pairs. Under this interpretation, the *Bellman equation* chooses and chains together one-step paths to find longer maximum reward paths.

Experimentally, we show how this simple reinterpration achieves equally good performance as a representative state-of-the-art method (HER) without using goal-rewards and vastly outperforms it in terms of reward sample complexity.

In this work, we also extend Floyd-Warshall Reinforcement Learning (FWRL) (Dhiman et al., 2018) to ~~the~~ use paramteric function approximators instead of tabular functions. We leverage FWRL's compositionality constraints in the space of GCVFs to introduce additional loss terms to the objective. However, these additional loss terms do not show improvement over the baseline. We conjecture that the compositionality constraints are already captured by other loss terms.

In summary, the contributions of this work are twofold. Firstly, we reinterpret goal-conditioned value functions as expected path-rewards and introduce one-step loss thereby removing their dependency on goal-rewards and reward-resampling. We showcase how the application of this interpreation leads to improved reward sample complexity. Secondly, we extend the tabular Floyd-Warshal Reinforcement Learning to use deep neural networks.

2 Background

An RL problem is formalized as an Markov Decision Process (MDP). A MDP governs an evolving sequence of state-action-reward triples $[(s_0, a_0, r_0), \dots, (s_T, a_T, r_T)]$, that is full governed by a four tuple $(\mathcal{S}, \mathcal{A}, T, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the system dynamics and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function In a typical RL problem the transition function T is not given but is known to be static. The objective of an RL problem is to find a policy $\pi(s) : \mathcal{S} \to \mathcal{A}$ that maximize the expected cumulative reward over time, called the returns $R_t = \sum_{k=t}^{\infty} r_k$. Reinforcement learning is has been typically formulated as in a single goal context, but more recently there has been interest in the multi-goal reinforcement learning problems.

2.1 Single-goal Reinforcement learning

A number of reinforcement learning algorithms use a parameteric function estimator for the expected future reward, either a value function $V(s; \theta)$ or an action-value function $Q(s, a; \theta)$. The more commonly used action-value function is defined as the running estimate of expected future reward

$$Q_\pi(s, a; \theta_Q) = \mathbb{E}_\pi \left[\sum_{k=t}^{T} \gamma^{k-t} R(s_k, a_k) \middle| s_t = s, a_t = a \right]. \tag{1}$$

When the policy π is optimal, the Q-function satisfies the *Bellman equation*:

$$Q_*(s_t, a_t; \theta_Q) = \begin{cases} R(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q_*(s_{t+1}, a; \theta_Q) & \text{if } t < T \\ R(s_T, a_T) & \text{otherwise} \end{cases}. \tag{2}$$

Once Q_*-function is approximated, a greedy policy can be computed from Q_*-function easily $\pi_*(s_t) = \arg\max_{a \in \mathcal{A}} Q_*(s_t, a)$.

Recent advances in deep reinforcement learning (Mnih et al., 2015a) view the Bellman equation as the gradient of a loss function. For example, Deep Q-Networks minimize the loss function whose gradient is the Bellman equation:

$$\mathcal{L}(\theta_{Q_m}, \theta_{\pi_m}) = \mathbb{E}_{a_t \sim \pi(s_t; \theta_{\pi_m})} \left[Q_m(s_t, a_t; \theta_{Q_m}) - R(s_t, a_t) - \max_a \gamma Q_t(s_{t+1}, a; \theta_{Q_t}) \right]^2, \tag{3}$$

where Q_m is the main network and Q_t is the target network. Target and main networks are function approximators with same structure, but different parameters. The target network parameters are changed slowly towards the main network parameters either by periodically copying the main network parameters to target network or by maintaining target network as exponential moving average of the main network. This improves the stability of the learning.

In this paper, we use a variation of Deep deterministic policy-gradients (DDPG) (Lillicrap et al., 2015) which extends DQN (Mnih et al., 2015a) to continuous action domains by introducing a parameterized policy function $\pi(s; \theta_\pi)$, which is learned through policy gradients.

2.2 Multi-goal Reinforcement learning

Multi-goal tasks need the specification of goal state which can change for each trial (Plappert et al., 2018a). The examples include navigation to a goal location, or moving a robotic arm so that the tip of the arm is at a particular 3D location (Fetch-Reach task).

The recent state of the art MGRL algorithms learn a goal-conditioned value function (GCVF), $Q\left(s,a,g\right)$, which is defined similar to the Q-function in Eq 4, but with an additional dependence over the desired goal specification $g \in \mathcal{G}$:

$$Q_{\pi}\left(s,a,g\right) = \mathbb{E}_{\pi}\left[\sum_{k=t}^{T} \gamma^{k-t} R(s_k, a_k, g) \middle| s_t = s, a_t = a\right]. \tag{4}$$

The goal specification $g \in \mathcal{G}$ can be arbitrary. To ease the learning of correspondence between the state and the "achieved goals", they are assumed to be an observable part of Goal-MDP $[(s_0, g_0, a_0, r_0), \dots, (s_T, g_T, a_T, r_T)]$.

Also note the dependence of reward function $R(s,a,g)$ on goal specification g. Hindsight Experience Replay (HER) builds upon this definition and stresses the importance of learning through episodes when the agent fails to reach the goal. HER works by resampling the trajectories from failed experiences and recomputing rewards on trajectories, as if the the achieved goals were the desired goals. As motivated earlier, however, this recompute requirement can be relaxed if the rewards are formulated independent of the goal specification.

Goal conditioned value function as path rewards

We define the goal-condition value function as expected path-rewards of reaching the goal from a given start state:

$$Q^P_{\pi}\left(s,a,g^*\right) = \begin{cases} \mathbb{E}_{l\in[t+1,T],\pi}\left[\sum_{k=t}^{l-1} \gamma^{k-t} R^P(s_k,a_k) \middle| s,a,g_l = g^*\right] & \text{if } \exists l \text{ such that } g_l = g^* \\ -\infty & \text{otherwise} \end{cases}. \tag{5}$$

The main difference between Q^P (5) and Q (4) is that Q^P explicitly depends upon reaching the goal within the episode, thereby removing the dependence of reward on the goal. Surprisingly, the Bellman Equation can still be applied to the path-value function Q^P with a slightly different terminal condition:

$$Q^P_{*}\left(s_t,a_t,g^*\right) = \begin{cases} R^P(s_t,a_t) + \gamma \max_{a\in\mathcal{A}} Q^P_{*}\left(s_{t+1},a,g^*\right) & \text{if } t < l-1 \text{ such that } g_l = g^* \\ R^P(s_{l-1},a_{l-1}) & \text{if } t = l-1 \text{ such that } g_l = g^* \end{cases}. \tag{6}$$

This formulation has several advantages. Firstly, it removes a design decision about the choice of goal-reward. Secondly, it enables HER to work without reward re-computation. This can be especially useful when reward computation is expensive.

Because we work with path rewards, the reward function design should make sure that there are no positive cycles in the environment. In another words, cycling back to the same state via another state should not yield positive rewards, otherwise the agent can extract unlimited reward just by going in circles. Hence a typical environment will have most of the reward values as negative. In all our experiments with path rewards, we use a constant value of -1 for all states, $R^P(s,a) = -1\, \forall s, a$.

3 Method

Since our new formulation of goal-conditioned value function satisifies the Bellman Equation, the existing MGRL algorithms for Q easily apply to the approximation of Q^P as long as we can estimate the terminal reward for Eq (5). The terminal reward, however, depends up on detecting whether the goal has been reached. To avoid this dependence, we introduce the step loss $\mathcal{L}_{\text{step}}$ which assumes that one-step reward is the highest reward between adjacent start-goal state:

$$\mathcal{L}_{\text{step}}(\theta_Q) = (Q^P_{*}\left(s_{l-1},a_{l-1},g_l;\theta_Q\right) - R(s_{l-1},a_{l-1}))^2. \tag{7}$$

This loss term allows us to estimate one-step path reward for all start and goal states. We modify an implementation of HER to include the step-loss term and disable goal rewards for our experiments. As in HER, we use the loss term from DDPG Lillicrap et al. (2015) while using "future" goal sampling strategy described in the paper. For this purpose an episode-wise replay buffer of transitions

$(s_t, g_t, a_t, r_t, s_{t+1}, g_{t+1}; g^*)$ is maintained. To draw samples from the replay buffer an episode and time step is chosen. The corresponding transition is taken to the sampled transition, however, the desired goal for this transition g^* is replaced by randomly chosen goal from the future transitions g_{t+f}. This new transition is applied to minimize the following DDPG loss function:

$$\mathcal{L}_{\text{ddpg}}(\theta_Q) = \left(Q_*^P\left(s_t, a_t, g_{t+f}; \theta_Q\right) - r_t - \gamma Q_t^P\left(s_{t+1}, \pi_t(s_{t+1}, g_{t+f}; \theta_\pi), g_{t+f}; \theta_Q\right)\right)^2 \tag{8}$$

3.1 Deep Floyd-Warshall Reinforcement Learning

We extend FWRL (Dhiman et al., 2018) to use deep neural network. The main contribution of FWRL is to employ compositionality constraints to exploit structure in the space of GCVFs. We translate these constraints into loss terms. In effect, the algorithm is same as the HER except addition of a few additional loss terms.

The constraint governing the FWRL algorithm states that the highest reward path from any state s_t to any goal g_{t+f} is greater than or equal to the sum of rewards via any intermediate state-goal pair (s_{im}, g_{im}):

$$Q_*\left(s_t, a_t, g_{im}\right) + Q_*\left(s_{im}, \pi_*(s_{im}, g_{t+f}; \theta_\pi), g_{t+f}\right) \geq Q_*\left(s_t, a_t, g_{t+f}\right), \tag{9}$$

where π_* is the optimal goal-conditioned policy and a_t represents the action taken at time t. Taking cue from Mnih et al. (2015a), we do not repeat the main online network Q_* in the loss term. We use target network Q_t and split the constraint into two loss terms. One loss term acts as a lower bound $\mathcal{L}_{\text{lo}}$ and the other acts as the upper bound $\mathcal{L}_{\text{up}}$:

$$\mathcal{L}_{\text{lo}} = \left(Q_t\left(s_t, a_t, g_{im}\right) + Q_t\left(s_{im}, \pi_t(s_{im}, g_{t+f}; \theta_\pi), g_{t+f}\right) - Q_*\left(s_t, a_t, g_{t+f}\right)\right)_+^2 \tag{10}$$

$$\mathcal{L}_{\text{up}} = \left(Q_*\left(s_t, a_t, g_{im}\right) + Q_t\left(s_{im}, \pi_t(s_{im}, g_{t+f}; \theta_\pi), g_{t+f}\right) - Q_t\left(s_t, a_t, g_{t+f}\right)\right)_+^2, \tag{11}$$

where $(\dots)_+$ denotes ReLU function. Note that the above terms differ only by choice of target network Q_t and main network Q_*.

Similar to HER we get the intermediate state by trajectory sampling. Once, a transition (s_t, a_t, r_t, s_{t+1}) and a future goal g_{t+f} has been sampled from the same episode, we sample another intermediate state and goal pair (s_{im}, g_{im}) such that $t \leq im \leq t + f$.

The resulting algorithm in shown in Alg 1.

4 Experiments

We use Fetch push, reach and pick and place tasks (Plappert et al., 2018a) in our experiments:

Fetch-Reach The tip of a robotic arm reaching a desired location.

Fetch-Push A robotic arm pushing a block to a desired location.

Fetch-Slide A robotic arm sliding a puck to a desired location.

4.1 Baseline: Hindsight experience replay

Hindsight Experience Replay Andrychowicz et al. (2016) targets goal-conditioned tasks. In goal-conditioned tasks, the rewards can be very sparse. Unless the agent hits the goal, no value-able reward is learned and rest of the space is almost even with respect to the rewards. To address this challenge, HER first assumes that for every s_t the achieved goal $g_t = f_g(s_t)$ is known. Then HER simulates as if the achieved goal g_t was the intended goal all along by re-sampling the goal conditioned reward function. More concretely, two time steps t_1 and $t_2 > t_1$ from the same episode in the replay memory are sampled. The achieved goal at t_2 , g_{t_2} is assumed to be the desired goal at t_1 and the new simulated transition becomes $(s_{t_1}, a_{t_1}, s_{t_1+1}, R(s_{t_1}, g_{t_2}, a_{t_1}))$, where $R(s_{t_1}, g_{t_2}, a_{t_1})$ is the recomputed reward instead of the observed reward $R(s_{t_1}, g_e^*, a_{t_1})$ that depends on the desired goal g_e^* for that episode e.

HER can be applied to either DDPG or DQN, the experiments in Andrychowicz et al. (2016) show them to be applied on DDPG.

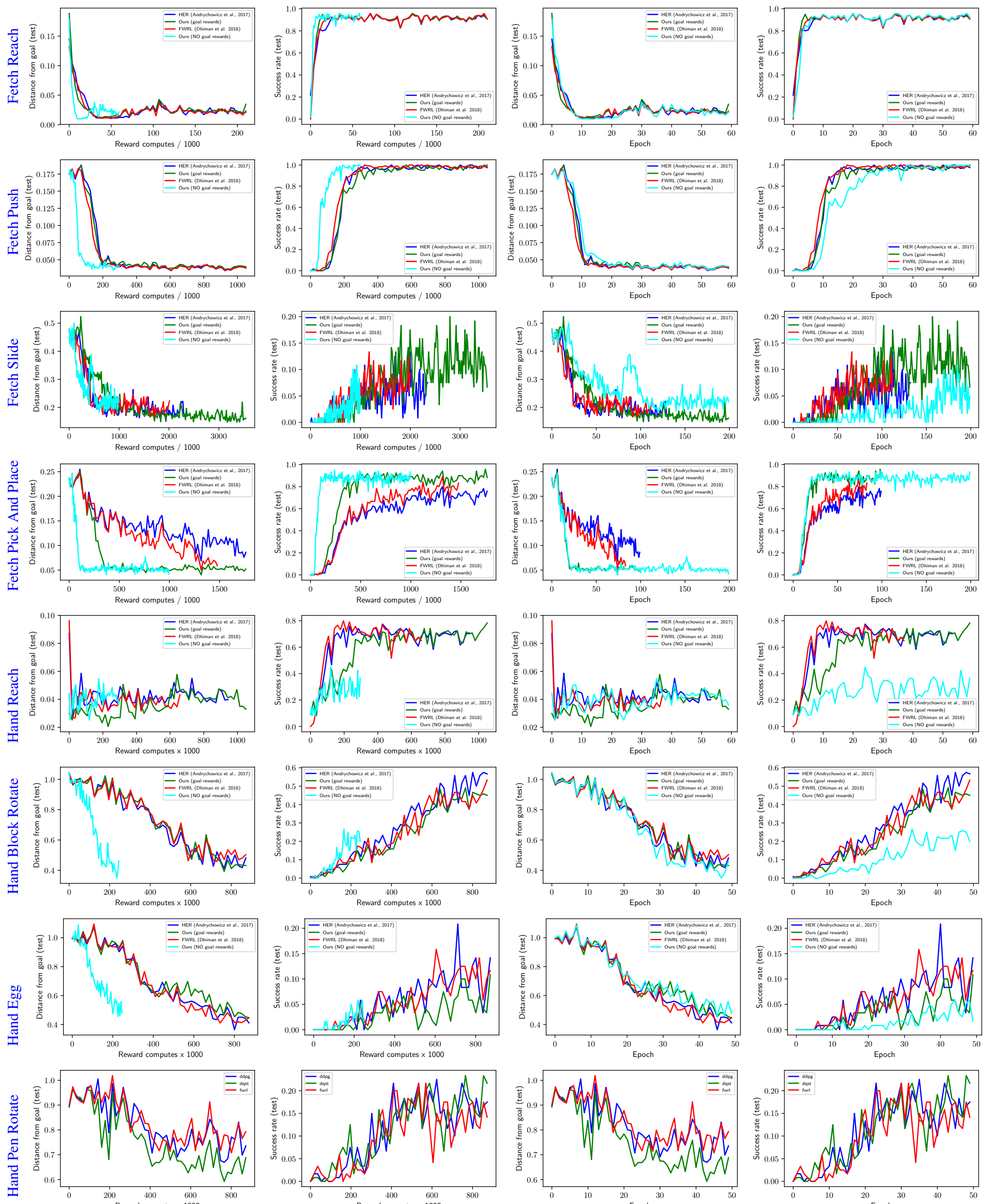


Figure 1: Effect of path reward on convergence in case of different loss functions on FetchReach. With path reward on PR-dqst ($\mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{step}}$) and PR-qlst ($\mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{step}} + \mathcal{L}_{\text{lo}} + \mathcal{L}_{\text{up}}$), are able achieve high success rate and even that takes longer than usual. Out of the two PR-dqst does better. The terms with PR use path-rewards and hence less computation by avoid recomputation of reward function. It is interesting that PR-dqst and PR-qlst reach closer in terms of distance from goal probably because of absence of threshold.

Algorithm 1: Path-reward reinforcement learning

```
/* By default all states are unreachable                                  */
Initialize networks $Q_* (s_i, a_i, g_j; \theta_Q)$ and $\pi(s_i, s_g; \theta_\pi)$ ;
Copy the main network to target network $Q_t (s_i, a_i; \theta_Q, s_j) \leftarrow Q_* (s_i, a_i; \theta_Q, s_j)$ ;
Initialize replay memory $M$ ;
/* Collect experience                                                     */
for $e \leftarrow 1$ to $M$ do
    Sample $g_e \in \mathcal{G}$ ;
    Set $t \leftarrow 0$;
    Observe state $s_t$ and achieved goal $g_t$ ;
    for $t \leftarrow 1$ to $T$ do
        Take action $a_t \leftarrow \epsilon$-greedy($\pi(s_t, g, Q)$) ;
        Observe $s_{t+1}, g_{t+1}, r_t$ ;
        Store $(s_t, g_t, a_t, s_{t+1}, g_{t+1}, r_t; g_e)$ in memory $M[e]$ ;
    /* Train                                                              */
    for $t \leftarrow 1$ to $T$ do
        HER sample batch
          $B = [(s_i, g_i, a_i, s_{i+1}, g_{i+1}, r_i; g_{i+f_i}), \dots, (s_b, g_b, a_b, s_{b+1}, g_{b+1}, r_b; g_{b+f_b})]$ from $M$ ;
        $\mathcal{L}(\dots) = 0$ ;
        for $b \in B$ do
            $(s_b, g_b, a_b, s_{b+1}, g_{b+1}, r_b, g_{b+f_b}) = B[b]$ ;
            $\mathcal{L}(\dots) += (Q_* (s_b, a_b, g_{b+1}) - r_b)^2$ ;          /* Step loss */
            $\mathcal{L}(\dots) += (Q_* (s_b, a_b, g_{b+f_b}) - r_b - \gamma Q_t (s_{b+1}, \pi_t(s_{b+1}, g_{b+f_b}; \theta_\pi), g_{b+f_b}))^2$
                /* DDPG loss                                                  */
        Update gradients for $Q_*$ and $\pi$ using loss $\mathcal{L}(\dots)$;
Result: $\pi^*(s_k, s_g, Q)$
```

5 Related Work

5.1 Experience Replay and Target Networks

The use of *experience replay* to learn from off-policy experiences and and slower-changing *target networks* are fundamental to the succesful operation of DFWRL. Both methods were popularized for use in the DRL literature in Mnih et al. (2015b)'s original DQN work.

5.2 Curriculum Learning

Curriculum learning, first introduced by Bengio et al. (2009), operates on the simple assumption that learning can in complex domains can be achieved by first iterating over simpler variants of the problem. This work builds of *Hindsight Experience Replay* (Andrychowicz et al., 2017), a form of *implicit curriculum* wherein past failed experiences are relabelled with goal states that in turn significantly accelerates learning.

5.3 Goal conditioned value functions

Several variants of goal-conditioned value functions have been explored extensively in the literature (Sutton et al., 2011; Schaul et al., 2015). Driven by the idea that simulataneously learning several tasks leads to better generalizability than learning a single tasks (Pong et al., 2018), goal-conditioned value functions have been shown to possess value in manipulation (Plappert et al., 2018b; Peng et al., 2018) and navigation domains (Zhang et al., 2017; Mirowski et al., 2018). Aside from Dhiman et. al., while much work has utilized these specialized value functions to learn how to solve complicated tasks, we were unable to find any other work that leveraged the structure in the space of these functions to accelerate and improve learning.

5.4 Multi-goal Reinforcement Learning Methods (HER/TDM)

Learning goal-conditioned value functions have also taked a variety of forms. In *model-free* RL, *Hindsight Experience Replay* (HER) (Andrychowicz et al., 2017) represents a form of *implicit curriculum* for training these functions. In HER, previous traversals are re-used for training with the goal state relabelled to be a part of these state traversals. HER sampling is shown to be highlight effective in training goal-conditioned value functions quickly and efficiently and is highly influential on our methods. In *model-based* RL, *Temporal Difference Models* (Pong et al., 2018) derive a connection between model-based and model-free algorithms for finite-horizon problems. TDM's define and learn goal and horizon-conditioned value functions and showcase improved performance over scores and sample efficiencies over model-based baselines while retaining the asymptotic performance of model-free RL. Being exclusively *model-free* and horizon-independent, DFWRL represents an orthogonal direction of research as compared to TDMs.

6 Conclusion

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Vikas Dhiman, Banerjee, Jeffrey M. Siskind, and Jason J. Corso. Floyd-warshall reinforcement learning: Learning from past experiences to reach new goals. *arXiv preprint arXiv:1809.09318*, 2018.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3389–3396. IEEE, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Piotr Mirowski, Matthew Koichi Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, and Raia Hadsell. Learning to navigate in cities without a map. *arXiv preprint arXiv:1804.00168*, 2018.

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015a.

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015b.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8. IEEE, 2018.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, abs/1802.09464, 2018a. URL `http://arxiv.org/abs/1802.09464`.

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

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

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

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 2371–2378. IEEE, 2017.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3357–3364. IEEE, 2017.

7 Appendix

7.1 Notation

Symbol	Meaning
$s \in \mathcal{S}$	State s in state space $\mathcal{S}$
$a \in \mathcal{A}$	Action a in Action space $\mathcal{A}$
$r \in \mathbb{R}$	Observed reward
$g \in \mathcal{G}$	Goal g in goal space $\mathcal{G}$
$f_g(s_t) : \mathcal{S} \to \mathcal{G}$	Function to compute achieved goal
$R(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$	Reward function
$R(s,g,a) : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$	Goal-conditioned Reward function
$T(s_t, a_t) \to s_{t+1}$	Transition function
γ	Discount factor
MDP=$(\mathcal{S}, \mathcal{A}, T, R, \gamma)$	Markov Decision Process
$\pi(s) : \mathcal{S} \to \mathcal{A}$	Policy function
$\pi(s,g) : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$	Goal conditioned Policy function
$Q_\pi(s,a;\theta_Q) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} R(s_k, a_k) \mid s_t = s, a_t = a]$	Q-function
$Q^*(s,a;\theta_Q) = \arg\max_\pi Q_\pi(s,a;\theta_Q)$	Optimal Q-function
$Q_\pi(s,a,g^*) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} R(s_k, g^*, a_k) \mid s_t = s, a_t = a]$	Goal conditioned Q-function

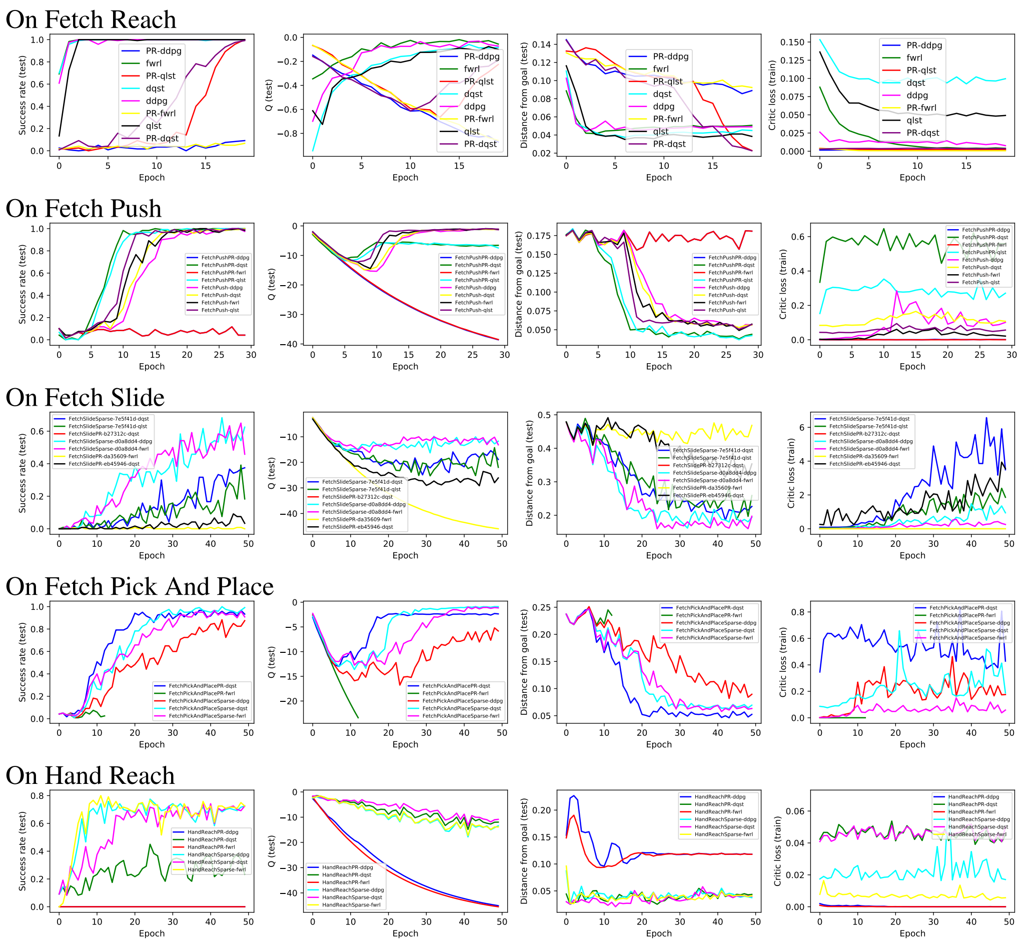


Figure 2: Effect of path reward on convergence in case of different loss functions on FetchReach. With path reward on PR-dqst ($\mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{step}}$) and PR-qlst ($\mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{step}} + \mathcal{L}_{\text{lo}} + \mathcal{L}_{\text{up}}$), are able achieve high success rate and even that takes longer than usual. Out of the two PR-dqst does better. The terms with PR use path-rewards and hence less computation by avoid recomputation of reward function. It is interesting that PR-dqst and PR-qlst reach closer in terms of distance from goal probably because of absence of threshold.

8 Old Experiments

We compared weighted combination of loss terms

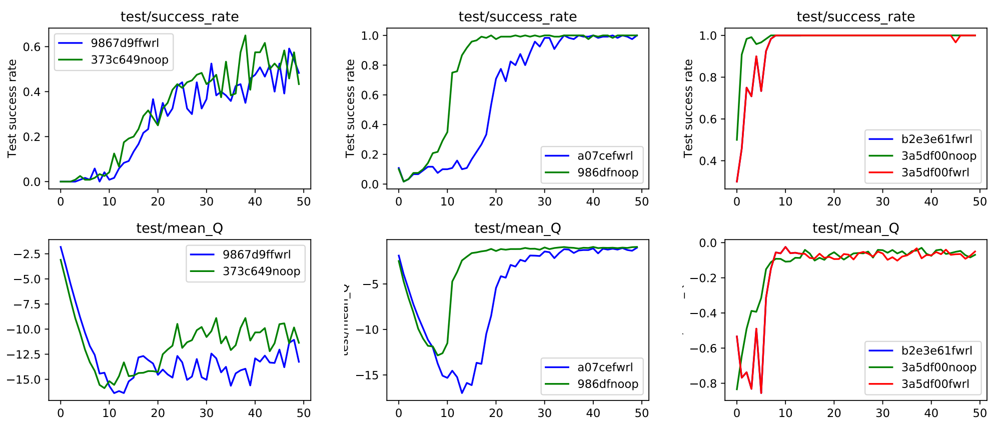
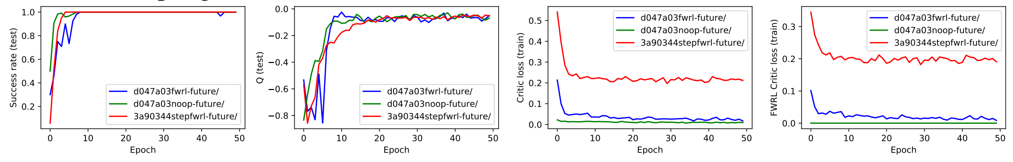
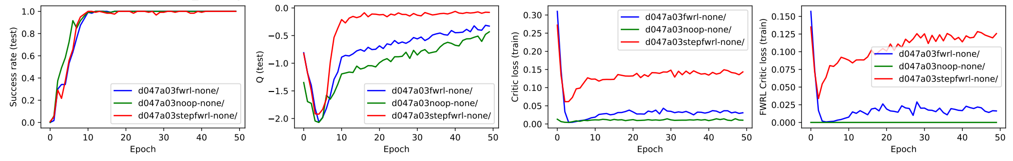


Figure 3: fwrl = Floyd Warshall ($= \mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{upper}}$) with HER sampling; noop = DDPG ($= \mathcal{L}_{\text{ddpg}}$)with HER sampling. Test success rate and Mean Q on (1) Fetch-Slide, (2) Fetch-Push and (3) Fetch-Reach task. fwrl does consistently worse than HER.

With HER sampling:



Without HER sampling:



Using both upper and lower bound in FWRL

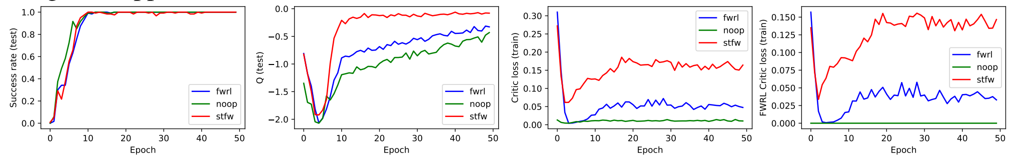


Figure 4: stepfwrl = DDPG loss $\mathcal{L}_{\text{ddpg}}$ + Step loss $\mathcal{L}_{\text{step}}$ + FWRL constraints $\mathcal{L}_{\text{upper}} + \mathcal{L}_{\text{lower}}$, noop = DDPG loss $\mathcal{L}_{\text{ddpg}}$ + HER sampling, fwrl = DDPG Loss $\mathcal{L}_{\text{ddpg}}$ + FWRL constraints $\mathcal{L}_{\text{upper}} + \mathcal{L}_{\text{lower}}$. All experiments on Fetch-Reach task.

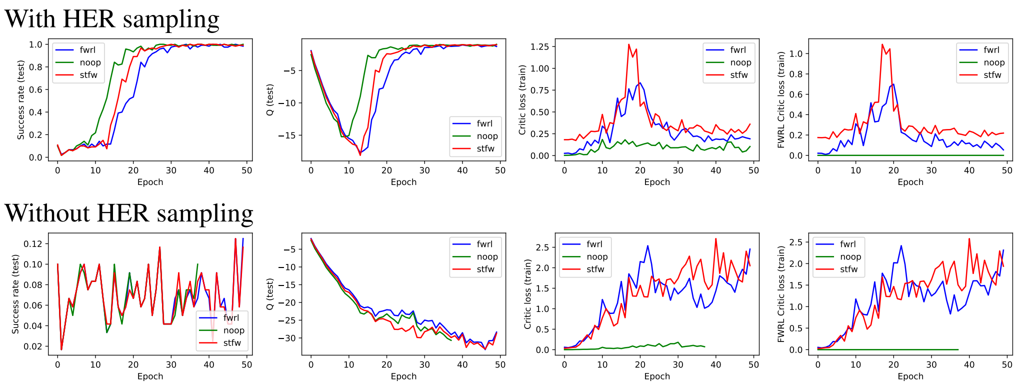


Figure 5: stepfwrl = DDPG loss $\mathcal{L}_{\text{ddpg}}$ + Step loss $\mathcal{L}_{\text{step}}$ + FWRL constraints $\mathcal{L}_{\text{upper}} + \mathcal{L}_{\text{lower}}$, noop = DDPG loss $\mathcal{L}_{\text{ddpg}}$ + HER sampling, fwrl = DDPG Loss $\mathcal{L}_{\text{ddpg}}$ + FWRL constraints $\mathcal{L}_{\text{upper}} + \mathcal{L}_{\text{lower}}$. All experiments on Fetch-Push

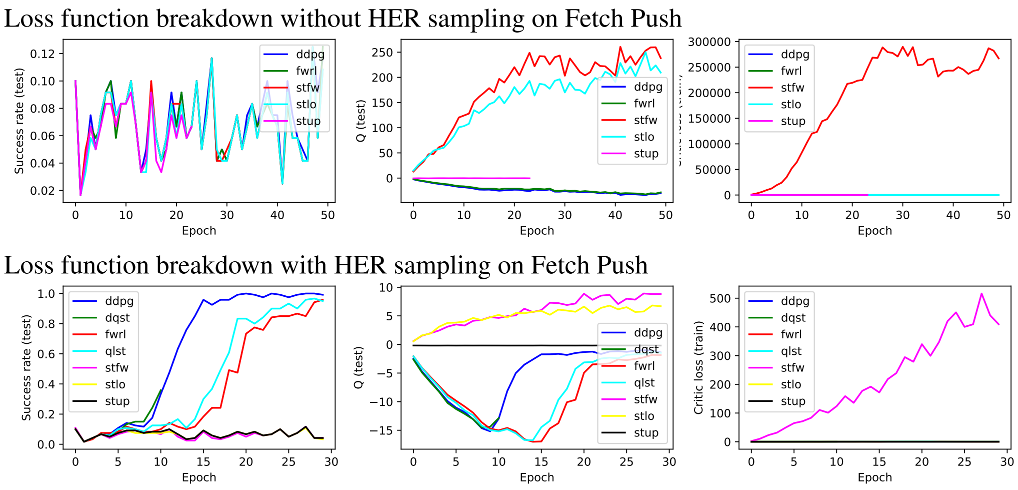
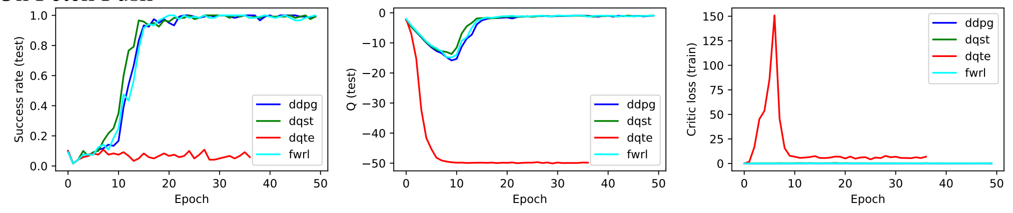
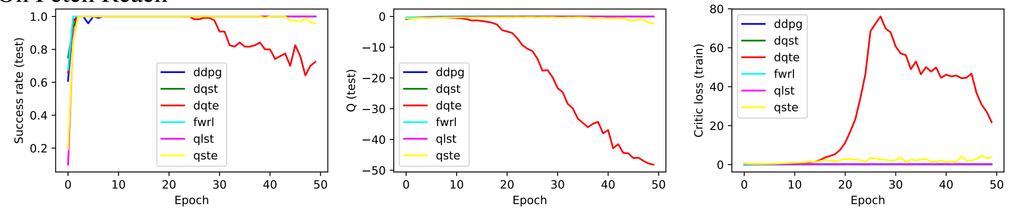


Figure 6: Fetch Push results. Loss function changes do no seem to make a difference. There are four parts to the loss function (1) DDPG Loss $\mathcal{L}_{\text{ddpg}}$, (2) Step loss$\mathcal{L}_{\text{step}}$, (3) Lower bound $\mathcal{L}_{\text{lower}}$ and (4) Upper bound $\mathcal{L}_{\text{upper}}$. ddpg = $\mathcal{L}_{\text{ddpg}}$, dqst = $\mathcal{L}_{\text{ddpg}}$ + $\mathcal{L}_{\text{step}}$, fwrl = $\mathcal{L}_{\text{ddpg}}$ + $\mathcal{L}_{\text{lower}}$ + $\mathcal{L}_{\text{upper}}$, qlst = $\mathcal{L}_{\text{ddpg}}$ + $\mathcal{L}_{\text{step}}$ + $\mathcal{L}_{\text{lower}}$ + $\mathcal{L}_{\text{upper}}$. stfw = $\mathcal{L}_{\text{step}}$ + $\mathcal{L}_{\text{lower}}$ + $\mathcal{L}_{\text{upper}}$, stlo = $\mathcal{L}_{\text{step}}$ + $\mathcal{L}_{\text{lower}}$, stup = $\mathcal{L}_{\text{step}}$ + $\mathcal{L}_{\text{upper}}$. Success rate is the fraction of times the agent reaches the goal. Q(test) is the estimated cumulative reward by the network. Critic loss is the total loss plotted during training. Because stfw, stlo, stup fail to succeed, we infer that the $\mathcal{L}_{\text{ddpg}}$ DDPG loss is critical for making the algorithm work. Since the qlst works better than fwrl, we infer that $\mathcal{L}_{\text{step}}$ Step loss is also important. only. Since there is slight improvement in dqst over ddpg, this means $\mathcal{L}_{\text{step}}$ really helps. dqst did not run fully but it shows promise (I need to fix a bug). But why does the loss for stfw keep rising? Does it mean that the SGD is not able to optimize the loss gradients in the right direction?

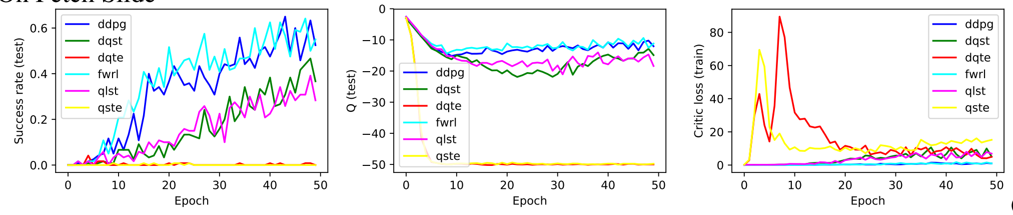
On Fetch Push



On Fetch Reach

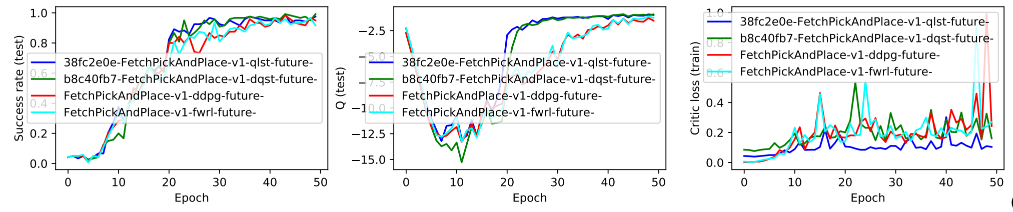


On Fetch Slide



On

Fetch Pick and Place



On

HandReach

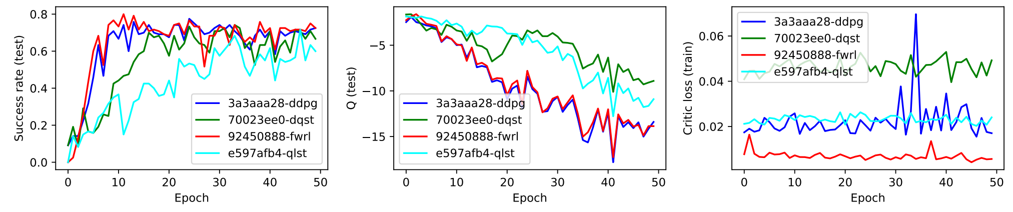


Figure 7: Fetch results. Loss function changes do no seem to make a difference. There are four parts to the loss function (1) DDPG Loss $\mathcal{L}_{\text{ddpg}}$, (2) Step loss$\mathcal{L}_{\text{step}}$, (3) Lower bound $\mathcal{L}_{\text{lo}}$ and (4) Upper bound $\mathcal{L}_{\text{up}}$. ddpg = $\mathcal{L}_{\text{ddpg}}$, dqst = $\mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{step}}$, fwrl = $\mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{lo}} + \mathcal{L}_{\text{up}}$, qlst = $\mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{step}} + \mathcal{L}_{\text{lo}} + \mathcal{L}_{\text{up}}$, dqte = $\mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{trieq}}$, qste = $\mathcal{L}_{\text{ddpg}} + \mathcal{L}_{\text{step}} + \mathcal{L}_{\text{trieq}}$. Success rate is the fraction of times the agent reaches the goal. Q(test) is the estimated cumulative reward by the network. Critic loss is the total loss plotted during training. Because stfw, stlo, stup fail to succeed, we infer that the $\mathcal{L}_{\text{ddpg}}$ DDPG loss is critical for making the algorithm work. Since the qlst works better than fwrl, we infer that $\mathcal{L}_{\text{step}}$ Step loss is also important. only. Since there is slight improvement in dqst over ddpg, this means $\mathcal{L}_{\text{step}}$ really helps. dqst did not run fully but it shows promise (I need to fix a bug). But why does the loss for stfw keep rising? Does it mean that the SGD is not able to optimize the loss gradients in the right direction?

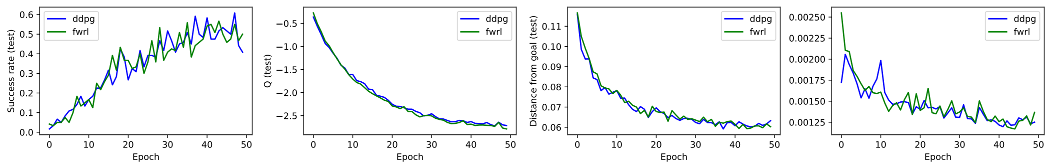


Figure 8: Experiment to see the effect of only path rewards on loss terms. We did not include a step term which becomes very important in this case.

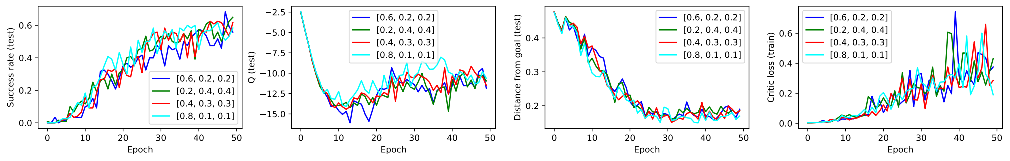


Figure 9: Effect of weighted combination of loss terms on FetchSlide. The three loss terms being weighed in order are $[\mathcal{L}_{\text{ddpg}}, \mathcal{L}_{\text{lo}}, \mathcal{L}_{\text{up}}]$

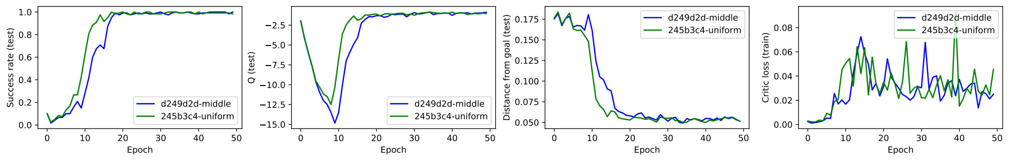


Figure 10: Effect of choosing the intermediate sample in the *middle* of the trajectory vs *uniform*ly random in the trajectory on FetchPush

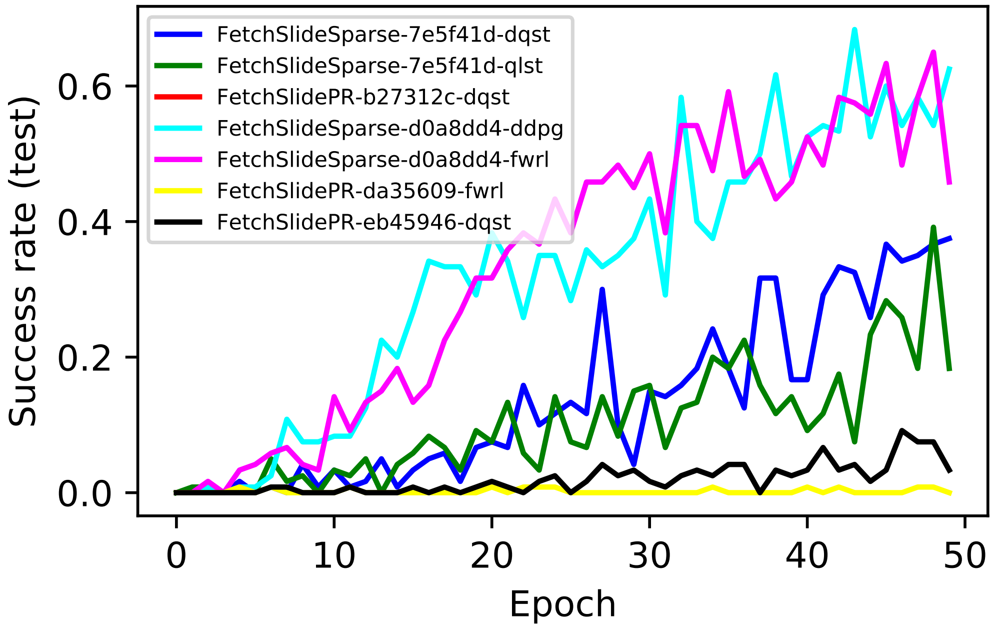


Figure 11: Effect of path rewards on FetchSlide