Intrinsically Motivated Multi-Goal Reinforcement Learning Using Robotics Environment Integrated with OpenAI Gym

Sivasubramanian Balasubramanian

SASTRA Deemed University, Tamil Nadu, India

Ex- Product Manager, Revalize, Real World Analytics

DOI: 10.55662/JST.2023.4502

ABSTRACT:

Sparse reward is one of the most challenging problems in reinforcement learning (RL). Hindsight Experience Replay (HER) attempts to address this issue by converting a failed experience to a successful one by relabelling the goals. In open-ended and changing environments, agents face a wide range of potential tasks that might not come with associated reward functions. Such autonomous learning agents must set their own tasks and build their own curriculum through an intrinsically motivated exploration. Because some tasks might prove easy and some impossible, agents must actively select which task to practice at any given moment, to maximize their overall mastery on the set of learnable tasks. The purpose of this technical report is two-fold. First, it introduces a suite of challenging continuous control tasks (integrated with OpenAI Gym) based on currently existing robotics hardware. The tasks include pushing, sliding and pick & place with a Fetch robotic arm as well as in-hand object manipulation with a Shadow Dexterous Hand. All tasks have sparse binary rewards and follow a Multi-Goal Reinforcement Learning (RL) framework in which an agent is told what to do using an additional input.

The second part of the paper presents a set of concrete research ideas for improving RL algorithms, most of which are related to Multi-Goal RL and Hindsight Experience Replay. The Fetch environments are based on the 7-DoF Fetch robotics arm,2 which has a two-fingered parallel gripper. Agents focus on achievable tasks first and focus back on tasks that are being forgotten. Experiments conducted in a new multi-task multi-goal robotic environment show that our algorithm benefits from these two ideas and demonstrate properties of robustness to distracting tasks, forgetting and changes in body properties

Keywords: *Deep Learning, Multi-Goal Reinforcement Learning, Robotics.*

1. INTRODUCTION:

Agents focus on achievable tasks first and focus back on tasks that are being forgotten. Experiments conducted in a new multi-task multi- goal robotic environment show that our algorithm benefits from these two ideas and demonstrate properties of robustness to distracting tasks, for- getting and changes in body properties.

Such autonomous learning agents must set their own tasks and build their own curriculum through an intrinsically motivated exploration. Because some tasks might prove easy and some impossible, agents must actively select which task to practice at any given moment, to maximize their overall mastery on the set of learnable tasks.

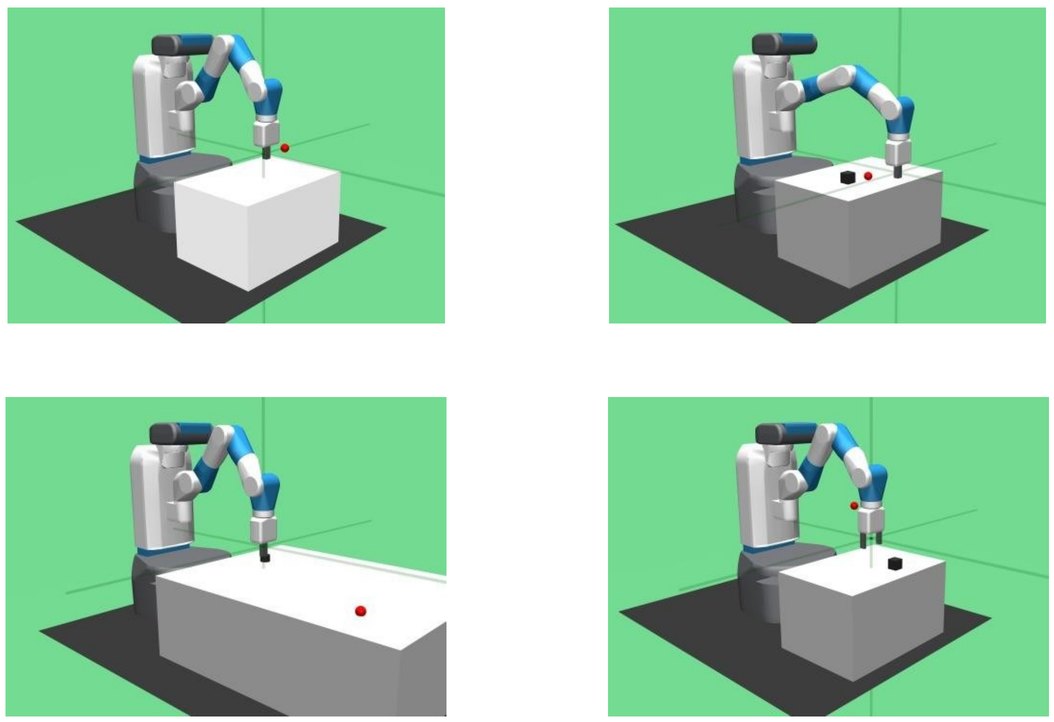
FETCH ENVIRONMENTS

In autonomous continual learning, agents must evolve in an ever-changing open-ended world and might face a variety of potential tasks. In such realistic environments, tasks cannot always be pre-specified by engineers. When the reward is sparse, deceptive, or even non-existing, the agent must be endowed with intrinsic motivations to explore the possibilities offered by its environment.

Reaching (FetchReach) The task is to move the gripper to a target position. This task is very easy to learn and is therefore a suitable benchmark to ensure that a new idea works at all.

Pushing (FetchPush) A box is placed on a table in front of the robot and the task is to move it to a target location on the table. The robot fingers are locked to prevent grasping. The learned behaviour is usually a mixture of pushing and rolling.

Figure 1.1: The four proposed Fetch environments: FetchReach, FetchPush, FetchSlide, and FetchPickAndPlace



Sliding (FetchSlide) A puck is placed on a long slippery table and the target position is outside of the robot's reach so that it must hit the puck with such a force that it slides and then stops at the target location due to friction.

Pick & Place (FetchPickAndPlace) The task is to grasp a box and move it to the target location which may be located on the table surface or in the air above it.

HAND ENVIRONMENTS

These environments based on the robot which is an anthropomorphic robotic hand with 24 degrees of freedom. Of those 24 joints, 20 can be controlled independently whereas the remaining ones are coupled joints. In all hand tasks, rewards are sparse and binary: The agent obtains a reward of 1 if the goal has been achieved (within some task-specific tolerance) and 0 otherwise. Actions are 20-dimensional: We use absolute position control for all non-coupled joints of the hand. We apply the same action in 20 subsequent simulator steps (with $\Delta t = 0.002$ each) before returning control to the agent, i.e. the agent's action frequency is $f = 25$ Hz.

Observations include the 24 positions and velocities of the robot's joints. In case of an object that is being manipulated, we also include its Cartesian position and rotation represented by a quaternion (hence 7-dimensional) as well as its linear and angular velocities. In the reaching task, we include the Cartesian position of all 5 fingertips.

Reaching (HandReach) A simple task in which the goal is 15-dimensional and contains the target Cartesian position of each fingertip of the hand. Similarly, to the FetchReach task, this task is relatively easy to learn. A goal is considered achieved if the mean distance between fingertips and their desired position is less than 1 cm.

Block manipulation (HandManipulateBlock) In the block manipulation task, a block is placed on the palm of the hand. The task is to then manipulate the block such that a target pose is achieved. The goal is 7-dimensional and includes the target position (in Cartesian coordinates) and target rotation (in quaternions). We include multiple variants with increasing levels of difficulty: HandManipulateBlockRotateZ Random target rotation around the z axis of the block. No target positions.

HandManipulateBlockRotateParallel Random target rotation around the z axis of the block and axis-aligned target rotations for the x and y axes. No target positions.

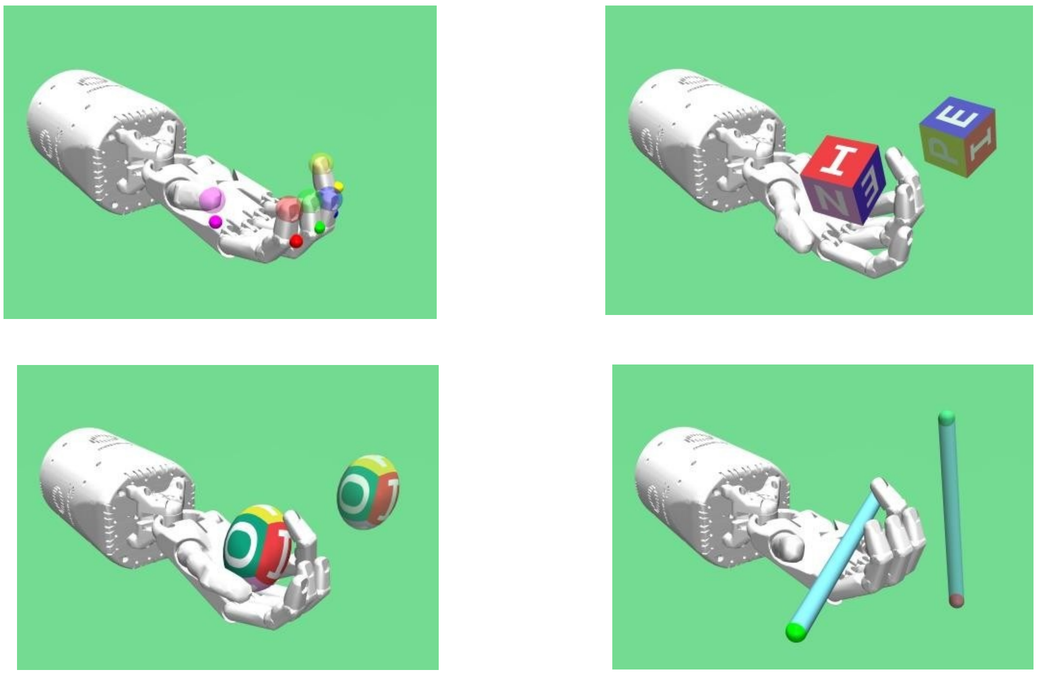
HandManipulateBlockRotateXYZ Random target rotation for all axes of the block. No target positions.

HandManipulateBlockFull Random target rotation for all axes of the block.

Random target position.

A goal is considered achieved if the distance between the block's position and its desired position is less than 1 cm (applicable only in the Full variant) and the difference in rotation is less than 0.1 rad.

Figure 2: The four proposed Shadow Dexterous Hand environments: HandReach, HandManipulateBlock, HandManipulateEgg, and HandManipulatePen



2. PROBLEM STATEMENT

We present CURIOUS, a multi-task and multi-goal reinforcement learning (RL) algorithm that uses intrinsic motivations to efficiently learn a finite set of multi-goal tasks in parallel. To build an algorithm able to learn multiple (multi-goal) tasks, one must answer the following questions:

1) How to choose the action policy architecture?
2) How to select the next task and goal to practice and learn about?
3) How to efficiently transfer knowledge between tasks and goals?

3. LITERATURE REVIEW

Hierarchical Extension: The idea of using a high-level policy to select goals for a lower-level policy was also stud- ied in the field of hierarchical RL. Yet, while hierarchical RL agents choose their own subgoals, they usually do so to achieve higher-level goals/tasks imposed by the engineer (Vezhnevets et al., 2017; Nachum et al., 2018; Levy et al., 2018). A natural extension of our work could be to replace our high-level MAB task selection policy by another CURI- OUS agent targeting self-generated higher-level tasks and goals, in a hierarchical manner

Learning a Goal Selection Policy: In this work we pro- vide the policy for goal selection: sampling uniformly from a pre-defined (reachable) goal space. In the future, the agents could learn it autonomously using adaptations of existing algorithms such as SAGG-RIAC (Baranes & Oudeyer, 2013) or GOAL-GAN (Held et al., 2017). SAGG-RIAC enables to split recursively a wide goal space and to focus on sub-regions where LP is higher, while GOAL-GAN pro- poses to generate goals of intermediate difficulty using a Generative Adversarial Network.

4. OTHER HAND ENVIRONMENTS

EGG MANIPULATION

HandManipulateEgg the objective here is similar to the block task but instead of a block an egg-shaped object is used. We find that the object geometry makes a significant difference in how hard the problem is and the egg is probably the easiest object. The goal is again 7-dimensional and includes the target position (in Cartesian coordinates) and target rotation (in quaternions). We include multiple variants with increasing levels of difficulty:

- HandManipulateEggRotate Random target rotation for all axes of the egg. No target positions.
- HandManipulateBlockFull Random target rotation for all axes of the egg. Random target position.

A goal is considered achieved if the distance between the egg's position and its desired position is less than 1 cm (applicable only in the Full variant) and the difference in rotation is less than 0.1 rad.

PEN MANIPULATION

HandManipulatePen Another manipulation, this time using a pen instead of a block or an egg. Grasping the pen is quite hard since it easily falls off the hand and can easily collide and get stuck between other fingers. The goal is 7-dimensional and includes the target position (in Cartesian coordinates) and target rotation (in quaternions). We include multiple variants with increasing levels of difficulty:

- HandManipulatePenRotate Random target rotation x and y axes of the pen and no target rotation around the z axis. No target positions.
- HandManipulateBlockFull Random target rotation x and y axes of the pen and no target rotation around the z axis. Random target position.

A goal is considered achieved if the distance between the pen's position and its desired position is less than 5 cm (applicable only in the Full variant) and the difference in rotation, ignoring the z axis,6 is less than 0.1 rad.

5. MULTI-GOAL ENVIRONMENT INTERFACE

All environments use goals that describe the desired outcome of a task. For example, in the FetchReach task, the desired target position is described by a 3-dimensional goal. While our environments are fully compatible with the OpenAI Gym API, we slightly extend upon it to support this new type of environment. All environments extend the newly introduced gym Goal Env.

GOAL AWARE OBSERVATION INTERFACE

First, it enforces a constraint on the observation space. More concretely, it requires that the observation space is of type gym Spaces. Dict space, with at least the following three keys:

- Observation: The actual observation of the environment, For example robot state and position of objects.
- Desired goal: The goal that the agent has to achieve. In case of FetchReach, this would be the 3-dimensional target position

- Achieved goal: The goal that the agent has currently achieved instead. In case of FetchReach, this is the position of the robot's end effector. Ideally, this would be the same as desired goal as quickly as possible.

EXPOSED REWARD FUNCTION

Second, we expose the reward function in a way that allows for re- computing the reward with different goals. This is a necessary requirement for HER-style algorithms which substitute goals. A detailed example is available in Appendix A

COMPATIBILITY WITH STANDARD RL ALGORITHMS

Since OpenAI Gym is commonly supported in most RL algorithm frameworks and tools like OpenAI Baselines, we include a simple wrapper that converts the new dictionary-based goal observation space into a more common array representation. A detailed example is available in Appendix A.

6. RESEARCH REQUEST

Deciding which problem is worth working on is probably the hardest part of doing research. Below we present a set of research problems which we believe can lead to widely-applicable RL improvements. For each problem we propose at least one potential solution but solving many of them will require inventing new ideas. To make tracking the progress of work on these ideas easier, we would like to ask authors to cite this report when publishing related research.

FASTER INFORMATION PROPOGATION

Most state-of-the-art off-policy RL algorithms use target net- works to stabilize training (e.g. DQN or DDPG). This, however, comes at a price of limiting the maximum learning speed of the algorithm as each target network update sends the information about returns only one step backward in time (if one-step boot- strapping is used). We noticed that the learning speed of DDPG+HER in the early stages of training is often proportional to the frequency of target network updates8 but excessive frequency/magnitude of target network updates leads to unstable training and worse final performance. How can we adapt the frequency of target network updates (or the moving average coefficient used to update the network) to maximize the training speed? Are there better ways to update the target network than a simple replacement or a moving average over time? Are there other ways to stabilize training which does not limit the learning speed (e.g. clipped objective similar to the one used in PPO).

RICHER VALUE FUNCTIONS

UVFA extended value functions to multiple goals, while TDM extended them to different time horizons. Both innovations can make training easier, even though the learned function is more complicated. What else could we fed to the value function to improve the sample-efficiency? How about discount factor or success threshold for binary rewards?

7. RESULTS AND DISCUSSIONS

AUTOMATIC HINDSIGHT GOALS GENERATION

The goals used for HER were generated using a hand-crafted heuristic, e.g. replaying with a goal which was achieved at a random future timestep in the episode. Instead, we could learn which goals are most valuable for replay. They could be chosen from the goals achieved or seen during training or generated by a separate neural network given a transition as input. The biggest question is how to judge which goals are most valuable for replay. One option would be to train the generator to maximize the Bellman error. This bears a lot of similarity to Prioritized Experience Replay and we expect that some techniques from this paper may be useful here

UNBIASED HER

HER changes the joint distribution of replayed (state, action, next_state, goal) tuples in an unprincipled way. This could, in theory, make training impossible in extremely stochastic environment albeit we have not noticed this in practice. Consider an environment in which there is a special action which takes the agent to a random state and the episode ends after that. Such an action would seem to be perfect in hindsight if we replay with the goal achieved by the agent in the future. How to avoid this problem? One potential approach would be to use importance sampling to cancel the sampling bias but this would probably lead to prohibitively high variance of the gradient.

ON-POLICY HER

HER generates data which is extremely off-policy and therefore multi- step returns cannot be used unless we employ some correction factors like importance sampling. While there are many solutions for dealing with off-policies of the data it is not clear if they would perform well in the setup where the training data is so far from being on-policy. Another approach would be to use multi-step optimality tightening inequalities. Using multi-step returns can be beneficial because the decreased frequency of bootstrapping can lead to less biased gradients. Moreover, it accelerates the transfer of information about the returns backwards in time which, accordingly to our experiment, is often the limiting factor in DDPG+HER training (compare previous paragraph).

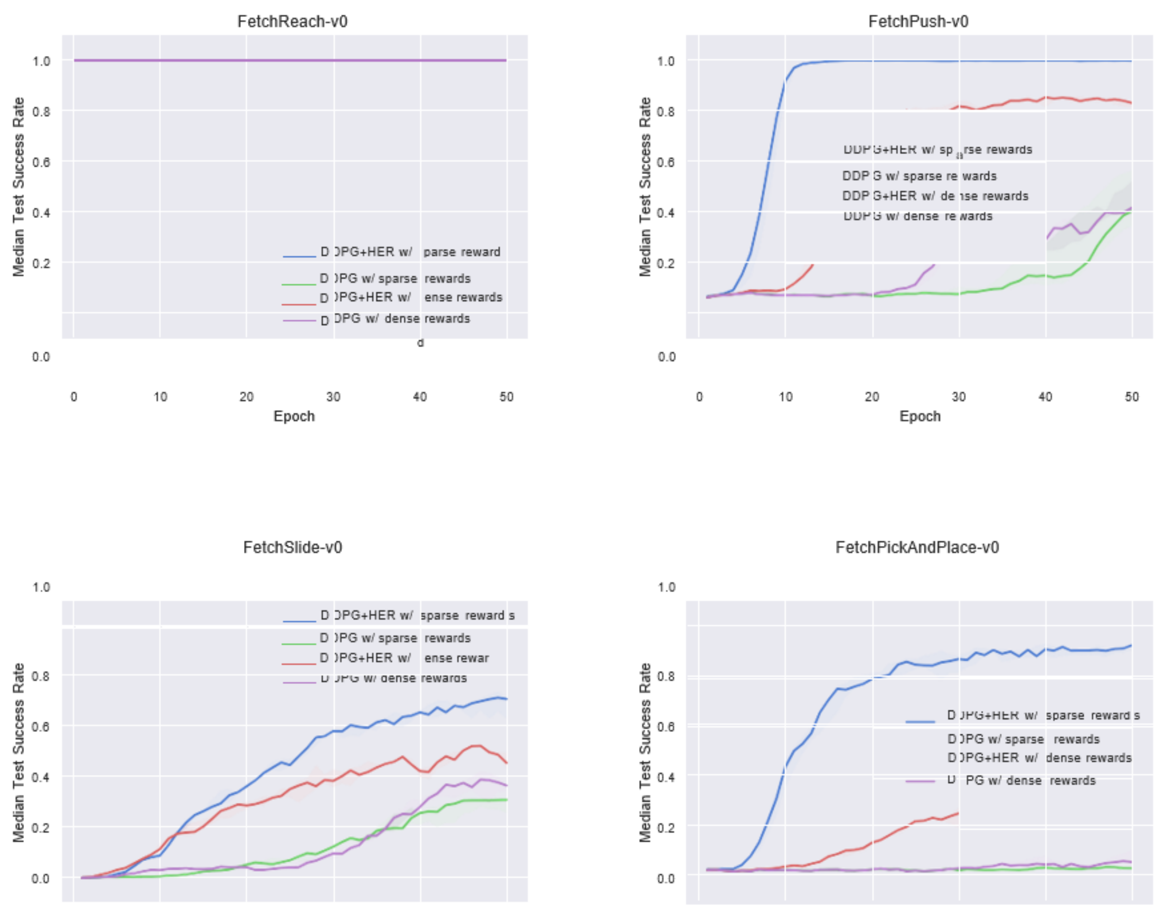


Figure 3: Median test success rate (line) with interquartile range (shaded area) for all four Fetch environments.

RL WITH FREQUENT ACTIONS

RL algorithms are very sensitive to the frequency of taking actions which is why frame skip technique is usually used on Atari. In continuous control domains, the performance goes to zero as the frequency of taking actions goes to infinity, which is caused by two factors: inconsistent exploration and the necessity to bootstrap more times to propagate information about returns backward in time. The problem of exploration can be addressed by using parameters noise for exploration and faster information propagation could be achieved by employing multi-step returns.

8. CONCLUSION AND FUTURE WORK

We assess the effectiveness of DDPG with and without Hindsight Experience Replay across all environments and their variants. We analyze four configurations for comparison, encompassing the evaluation of performance in different scenarios.

1) DDPG+HER with sparse rewards
2) DDPG+HER with dense rewards
3) DDPG with sparse rewards
4) DDPG with dense rewards

For all environments, we train on a single machine with 19 CPU cores. Each core generates experience using two parallel rollouts and uses MPI for synchronization. For FetchReach, FetchPush, FetchSlide, HandReach and , FetchPickAndPlace we train for 50 epochs (one epoch consists of 50 full episodes), which amounts to a total of 4.75 ·106 timesteps. For the remaining environments, we train for 200 epochs, which amounts to a total of 38 106 timesteps. We evaluate the performance after each epoch by performing 10 deterministic test rollouts per MPI worker and then compute the test success rate by averaging across rollouts and MPI workers. Our implementation is available as part of OpenAI Baselines7. In all cases, we repeat an experiment with 5 different random seeds and report results by computing the median test success rate as well as the interquartile range.

Figure 3 depicts the median test success rate for all four Fetch environments. FetchReach is clearly a very simple environment and can easily be solved by all four configurations. On the remaining environments, DDPG+HER clearly outperforms all other configurations. Interestingly, DDPG+HER performs best if the reward structure is sparse but is also able to successfully learn from dense rewards. For vanilla DDPG, it is typically easier to learn from dense rewards with sparse rewards being more challenging.

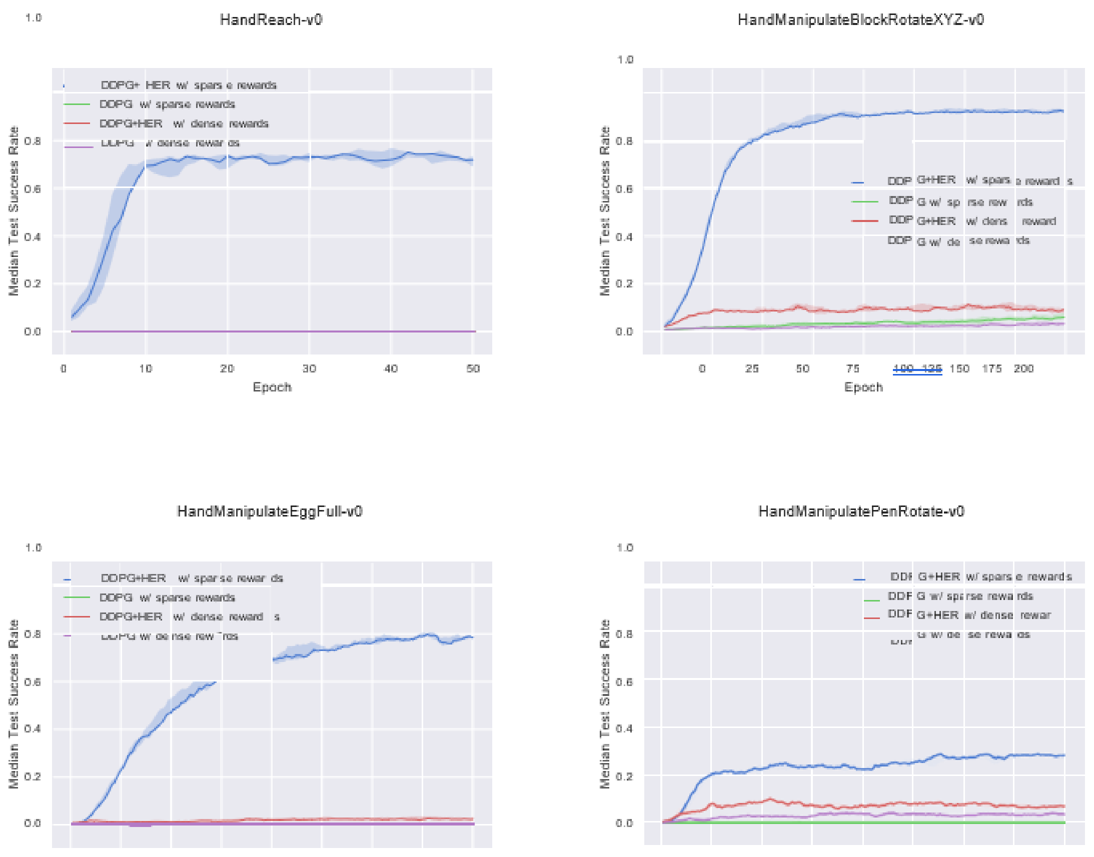


Figure 4: Median test success rate (line) with interquartile range (shaded area) for all four Fetch environments.

This depicts the median test success rate for all four hand environments. Like the Fetch environments, DDPG+HER significantly outperforms the DDPG baseline. In fact, the baseline often is not able to learn the problem at all. Like before, the sparse reward structure works significantly better than the dense reward when using HER. HER can learn partly successful policies on all environments but especially HandManipulatePen is especially challenging and we are not able to fully solve it. Note that we do not depict results for all variants of the four environments in this figure. A complete set of plots for all environments and their variants can be found.

We believe the reason why DDPG+HER typically performs better with sparse rewards is mainly due to the following two reasons:

Learning the critic is much simpler for sparse rewards.

- In the dense case, the critic has to approximate a highly non-linear function involving the Euclidean distance between positions and the difference between two quaternions for rotations. Conversely, learning the sparse return is much simpler, as the critic only needs to distinguish between successful and failed states.
- A dense reward biases the policy towards a specific strategy. For example, it might be advantageous to initially grasp an object properly and then start rotating it towards the desired goal. However, the dense reward encourages the policy to choose a strategy that directly achieves the desired goal.

Deciding which problem is worth working on is probably the hardest part of doing research. Below we present a set of research problems which we believe can lead to widely-applicable RL improvements. For each problem we propose at least one potential solution but solving many of them will require inventing new ideas. To make tracking the progress of work on these ideas easier, we would like to ask authors to cite this report when publishing related research.

9. REFERENCES

i. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In Advances in Neural Information Processing Systems, pages 5055–5065.

ii. Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. arXiv preprint arXiv:1707.06887.

iii. Brockman,G.,Cheung,V.,Pettersson,L.,Schneider,J.,Schulman,J.,Tang,J.,andZaremba, W.(2016). Openai gym. arXiv preprint arXiv:1606.01540.

iv. Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. (2017). OpenAI Baselines.

v. Florensa, C., Held, D., Wulfmeier, M., and Abbeel, P. (2017). Reverse curriculum generation for reinforcement learning. arXiv preprint arXiv:1707.05300.

vi. Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., Schölkopf, B., and Levine, S. (2017). Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. arXiv preprint arXiv:1706.00387.

vii. He, F. S., Liu, Y., Schwing, A. G., and Peng, J. (2016). Learning to play in a day: Faster deep reinforcement learning by optimality tightening. arXiv preprint arXiv:1611.01606.

viii. Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. A Novel Approach for Color Image, Steganography using NUBASI and Randomized, Secret Sharing Algorithm. Indian Journal Science and Technology, 8(S7), 228-235.