Follow your Nose: Using General Value Functions for Directed Exploration in Reinforcement Learning

Somjit Nath[1], **Omkar Shelke**[1], **Durgesh Kalwar**[1], **Hardik Meisheri**[1], **Harshad Khadilkar**[1, 2]

[1]TCS Research
[2]IIT Bombay

Abstract

Exploration versus exploitation dilemma is a significant problem in reinforcement learning (RL), particularly in complex environments with large state space and sparse rewards. When optimizing for a particular goal, running simple smaller tasks can often be a good way to learn additional information about the environment. Exploration methods have been used to sample better trajectories from the environment for improved performance while auxiliary tasks have been incorporated generally where the reward is sparse. If there is little reward signal available, the agent should employ exploration strategies to reach parts of state space for sub-goal identification it can then optimize towards. However, that exploration needs to be balanced with the need for exploiting the learned policy. Hence we use 'directed' exploration. This paper explores the idea of combining exploration with auxiliary task learning with General Value Functions (GVFs).

Introduction

Reinforcement Learning (RL) has shown great advances in the recent years, in solving sequential decision making tasks. Particularly in the field of games, Reinforcement Learning algorithms have managed to achieve superhuman levels of performance(Mnih et al. 2013) (Silver et al. 2016). In reinforcement learning, the problem of exploration vs exploitation is a persistent problem, particularly in complicated environments. It is a common practice to use ϵ-greedy strategies for exploration and that is what most algorithms use generally. However, for complicated environments for example in the game of Montezuma's Revenge in Atari (Mnih et al. 2013), the RL agent is unable to learn anything because the state space for fully random exploration is too large.

In Reinforcement Learning, the main goal is to obtain a policy, π that maximises the expected discounted return

$$\max_{\pi} \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t)]$$

from interaction with the environment in the form of states s_t, actions a_t, and rewards r_t. At the beginning, the agent will not have much idea about the environment it is in and thus needs to gather information before it can take useful actions. If the state space is very complicated and/or the reward is very sparse, obtaining useful trajectories (collection of states, actions and rewards) can be difficult. This is the problem of exploration in RL which is still quite an open area for research.

There have been many recent papers that address the problem of exploration like Noisy Networks (Fortunato et al. 2017), Count-based Exploration (Tang et al. 2017), Curiosity Driven Exploration (Pathak et al. 2017), unsupervised learning of goal space for intrinsically motivated exploration (Péré et al. 2018), Go-explore (Ecoffet et al. 2019) which goes to a previously explored state and restarts exploration from that state. One interesting breakthrough for solving hard exploration problems came in the Never Give Up (Badia et al. 2020b) paper, which was included in the Agent 57 (Badia et al. 2020a) paper which is able to successfully solve all the 57 Atari games. This paper introduces two additional rewards for defining directed exploration, the episodic and the intrinsic reward. Agent 57 uses a family of policies with varying degree of exploration trained by Recurrent Experience Replay in Distributed Reinforcement Learning (R2D2) (Revaud et al. 2019) with Retrace (λ) (Munos et al. 2016) to account for off-policy learning. Most of these methods require learning some form of novelty or generating a reward function based on curiosity. None of them are as simple as the traditional ϵ-greedy approach. Recently, (Dabney, Ostrovski, and Barreto 2020) came up with a temporally extended version of the ϵ-greedy exploration strategy that can handle complicated environments as well.

In our paper, we extend upon this strategy by using auxiliary task learning with the help of General Value Functions to perform directed exploration thereby further improving state space coverage during exploration. We thus have an exploration strategy based that augments on the temporally extend ϵ-greedy exploration with sub-policies from the General Value Functions to perform better exploration. This reformulation also has major advantages, particularly with respect to representation learning.

Background

Temporally Extended EZ-greedy Exploration: In generic ϵ-greedy exploration, the agent can either take an action that maximizes its own state-action values with probability 1-ϵ or a random action with probability ϵ. Thus given infinite

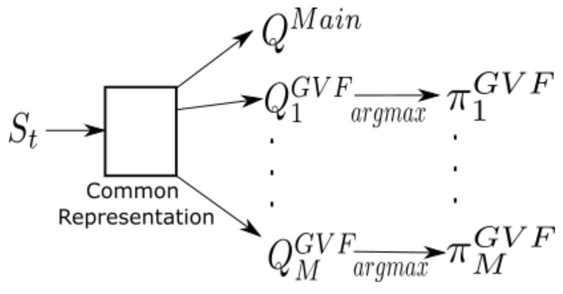


Figure 1: This figure represents the architecture used for DEZ-greedy algorithm with M GVFs having M policies

time, it would explore the entirety of the state space as it would be taking random actions infinite times provided we always have a non-zero ϵ always. This form of exploration is generally very slow in covering different parts of the state space. Thus temporally extended ϵ-greedy (Dabney, Ostrovski, and Barreto 2020) fixes a random action and keeps taking the action for a certain number of time steps, called persistence (Z) of the random action. So instead of taking one random step with probability ϵ, it executes a sequence of same random actions. The persistence Z is not a constant value throughout the exploration process but it is uniformly sampled from 0 i.e. no exploration to a maximum number of time-steps (a hyper-parameter, which would depend on the environment). This is why the algorithm is also known as EZ greedy.

In the paper, the authors mostly used the same random action repeated K times, but we can easily replace a sequence of repeated actions with a hand-crafted or learned option (Sutton, Precup, and Singh 1999). These options could help to reach parts of state-space that could be novel or would help in solving the main goal. In this paper, we formulate a framework to learn such options by using General Value Functions.

General Value Functions: A value function provides an estimate of the discounted total expected return that can be obtained given a certain state-action pair and a policy π. Similarly, General Value Functions (GVFs) (Sutton et al. 2011) can provide an estimate of environment knowledge given it can optimize over a certain reward function given a policy and a discount factor. Thus, if the discounted returns be G_t and for a trajectory length T,

$$Q^{GVF}(s, a; \pi, \gamma, r) = \mathbb{E}[G_t | s_t, a_t, A_{t+1:T-1} \sim \pi]$$

The input to the GVFs would thus be a policy, discount, reward function (also called cumulants) and these can be described as questions and for particular question functions, GVFs can be trained with normal RL algorithms (like Q-learning) and thus can provide useful information about the environment and the transition dynamics. In the Horde architecture (Sutton et al. 2011), the authors show the usefulness for answering predictive questions about the environment as well as how it can learn multiple control policies from a single behaviour policy.

The cumulants can also be viewed as providing auxiliary tasks the agent needs to perform before it can solve the main task. Often such auxiliary tasks would have no reward signal associated with them otherwise. Hand-crafted cumulants can help in learning (Jaderberg et al. 2016), but the agent must be able to learn GVF questions on its own as formulated in (Veeriah et al. 2019). In this paper, we learn the GVFs off-policy i.e. they are learned with respect to its own greedy policy. Thus, maximising for Q^{GVF} generates a different sub-task oriented policy which can be viewed as an option, to be used for a short persistence period. Thus, by following the policy for a short period we can generate useful options.

Algorithm 1: DEZ-greedy exploration strategy

```
Function DEZGreedy($\epsilon$, $Z_{max}$):
    $z \leftarrow 0$
    $w \leftarrow -1$
    $g \leftarrow 0$
    while True do
        Observe state $s$
        if $z == 0$ then
            if $random() < \epsilon$ then
                Sample duration: $z \sim [1, Z_{max}]$
                Sample GVF: $g \sim [0, M]$
                if $g == 0$ then
                    Sample action: $w \leftarrow U(A)$
                    $a \leftarrow w$
                else
                    $a \leftarrow argmax(Q_g^{GVF})$
            else
                $a \leftarrow argmax(Q^{Main})$
        else
            if $g == 0$ then
                $a \leftarrow w$
            else
                $a \leftarrow argmax(Q_g^{GVF})$
            $z \leftarrow z - 1$
        Take Action $a$
```

Directed EZ Greedy Algorithm

Directed EZ Greedy learns auxiliary tasks in the form of GVFs and uses sub-policies obtained by greedy action selection from the General Value Functions learned as options for temporally extended ϵ-greedy algorithm. With probability $epsilon$ it selects a random option which can be sub-policies from any of the randomly chosen GVFs at that particular state and keeps taking the greedy action with respect to that GVF. Thus, it explores different parts of the state space by following a specific GVF policy which makes the exploration directed, hence the name Directed EZ Greedy. The entire algorithm is portrayed in Algorithm 1.

The algorithm learns M+1 Value Functions as shown in Figure 1 where M is the number of GVFs. These Value Functions are learned from a shared common representation through which the losses for both the main Q values and the GVFs propagate. In Algorithm 1, we sample an action from M+1 ([0, M]) possible options. In all our experiments, we

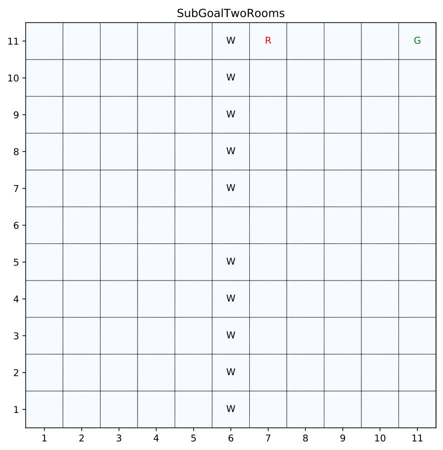


Figure 2: SubGoal Two Rooms Environment

include the option of repeating a random action as one additional option as well.

There are 3 main advantages of this approach:

1. There is no requirement for handcrafted options, rather DEZ-greedy can take sub-polices defined by the GVFs. Thus the input to generate the options can be as simple is defining a reward and a termination function as opposed to a sequence of actions.
2. Normally GVFs have been used to give predictive knowledge of the environment and this directly helps in learning good representations. In Figure 1 as well, we employ a similar structure whereby the GVFs not only influence the exploration policies but also help in learning better representations.
3. Learning GVFs while following the behavior policy leads to divergence if we do not use importance sampling ratios as the greedy policy of the GVFs will be off-policy. However following the GVF policy initially helps to keep the behavior policy closer to the greedy GVF policy (at least during the exploration phase). Following that even if the GVFs diverge, we do not use it explicitly because we anneal ϵ gradually. This further reduces the impact of off-policy divergence.

Environments

Most of our experiments were tested in the gridworld setting with hand-crafted cumulants. We mainly considered two environments:

- **Two Rooms Environment:** Here, the goal of the agent is to reach the adjacent room by passing through a corridor. As shown in Figure 2, the agent spawns in the bottom left corner and it has to find its way to the green dot on the top right. This seems like an easy task, but for larger dimensions it becomes difficult as the agent has to explore for a long time before it reaches the goal. The agent receives the (x,y) coordinates of its position as its state. The agent gets a reward of +1 on reaching the Green dot and a step reward of -0.01 with maximum time step of 300 after which the episode ends. For this environment, we used only one GVF which gets a cumulant of 1 for crossing the corridor.
- **SubGoal Two Rooms Environment:** This is a modification of the Two Rooms Environment with similar dynamics, except the agent has to collect the red dot and then go to the green dot. The reward for reaching the green dot without collecting the red dot is +0.1, whereas it is +1 after collecting the red dot. The other parameters are exactly the same. For this environment, we append a boolean value of whether the agent has visited the red flag to the (x,y) coordinates to make the states Markov. In this environment, we used one additional GVF, which receives a cumulant of +1 if the agent visits the red dot.

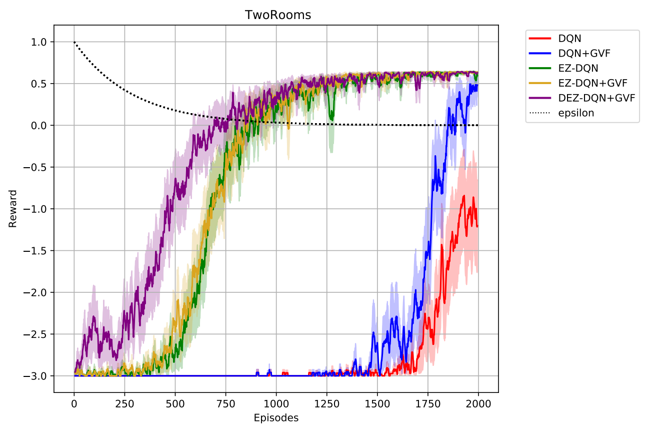
Results

Algorithms compared: For our experiments, we compare 5 different algorithms.

- The RL algorithm used for our experiments is the DQN algorithm (Mnih et al. 2013), i.e. all the GVFs and the main Q values are learned using DQN. The baseline to compare against would thus be DQN. DQN uses ϵ-greedy exploration with annealing ϵ and that is what we used for this experiment as well.
- EZ-DQN is the temporally extended ϵ-greedy (Dabney, Ostrovski, and Barreto 2020) version of DQN with EZ-greedy exploration.
- DQN+GVF is the algorithm that uses GVFs only for representation learning via the common representation which is modified by both the main DQN agent as well as all the GVFs. This still uses ϵ-greedy exploration like DQN.
- EZ-DQN+GVF uses the exact architecture as above except the behaviour policy uses EZ-greedy exploration strategy.
- DEZ-DQN+GVF is our algorithm that uses GVFs not only for learning better representations but also for learning options which can help in directed exploration.

For all the algorithms we used an annealing ϵ strategy as shown in Figure 3 and Z_{max} of 30. The Figure 3 portrays the learning curves of the algorithms across 10 runs over 2000 episodes. The detailed hyper-parameters are listed in Table 1.

From Figure 3 it is evident that EZ-greedy helps a lot in exploring the state space much much quickly and there is a marked difference between algorithms that use it and normal ϵ-greedy. However, adding directed exploration with the use of GVFs makes the RL agent learn even faster because of exploration of useful state-spaces as the GVFs are designed to stimulate goal-driven exploratory behaviour. Another interesting thing worth noting is that even adding GVFs without using any exploratory strategy helps in better performance since the DQN+GVF (blue) is slightly better than DQN (red). Even in the case of using EZ-greedy with GVFs there is an improvement in performance in the SubGoal Two



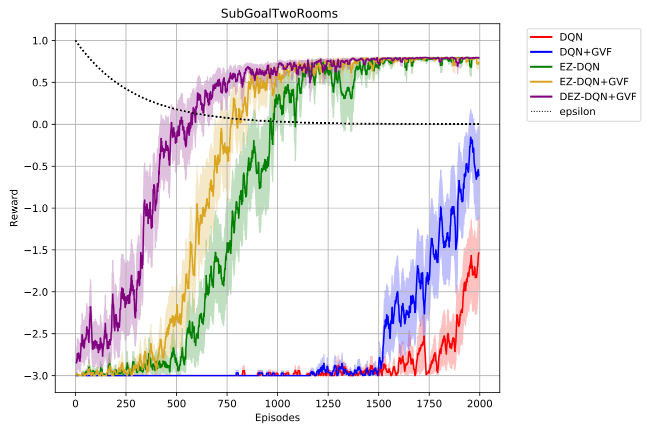


Figure 3: Different Algorithms compared on 19x19 TwoRooms and 11x11 SubGoal Two Rooms Environment

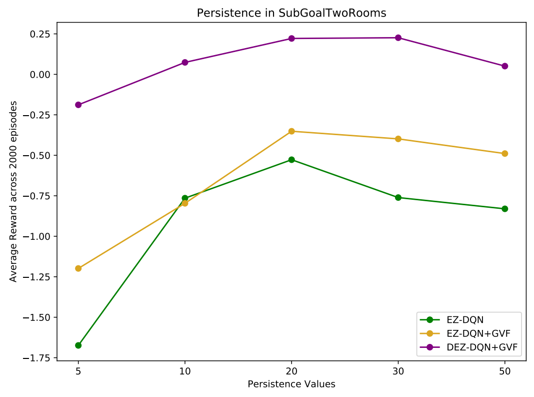


Figure 4: Parameter Sensitivity of EZ-greedy and DEZ-greedy strategies with respect to the persistence, Z_{max} in SubGoal Two Rooms Environment

Rooms environment. This is likely due to GVFs learning useful representations which can improve performance.

Another aspect that is crucial while using GVFs is off-policy learning. Learning GVFs on-policy with respect to the behaviour policy might not yield as useful options (sub-policies) as learning them off-policy. In the DEZ-greedy exploration strategy since we sample actions greedily from the GVFs often, the resulting behaviour policy is closer to the greedy policy of the GVFs. This is particularly true in our case as most of our GVFs are goal-driven.

Figure 5 shows us the Q values for each of the individual actions for the GVF that gets a reward of +1 on collecting the red dot in the SubGoal Two Rooms Environment. It is evident that for EZ-DQN+GVF, the best performing algorithm apart from ours, has divergent Q values, since we never sample any actions greedily from the GVFs, the GVFs diverge to non-sensical values. However, the values for DEZ-DQN+GVF are much well inside bound. In addition, following the optimal policy of this GVF will lead the agent into places close to the red dot thereby increasing the chances of picking up the red dot during exploration.

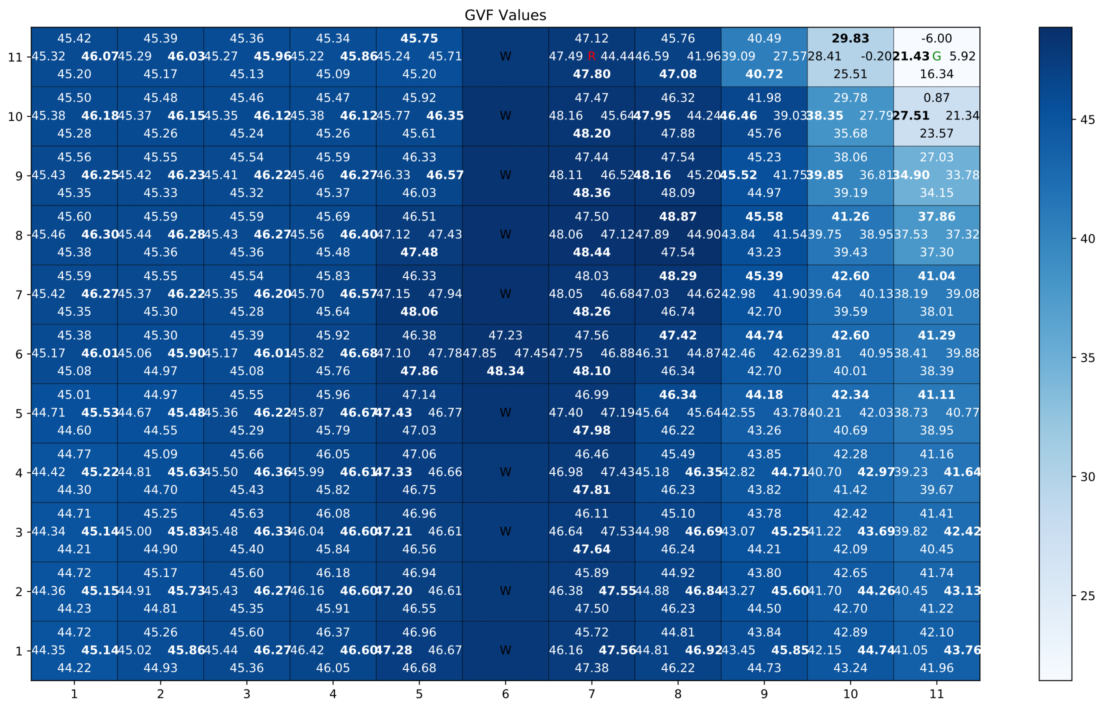
The performance of the algorithm will be determined by the persistence values chosen. For example, a persistence value of 1 for EZ exploration reduces to ϵ greedy exploration. In Figure 4, we plot the sensitivity of the algorithms across different persistence values. For smaller values the performance degrades due to less exploration, however, for directed exploration, the drop in performance is significantly less as even with smaller persistence values the agent can construct meaningful options which makes DEZ-greedy much more robust.

Exploration is one of the critical aspects for good performance for Reinforcement Learning, especially in games. EZ-greedy (Dabney, Ostrovski, and Barreto 2020) has been shown to perform better overall in a variety of Atari games compared to curiosity driven methods which specialize in exploration for some specific games. DEZ-greedy incorporates aspects of both EZ-greedy and auxiliary task learning along with learning richer representations and is more robust as well. Thus, it should work well in a variety of games.

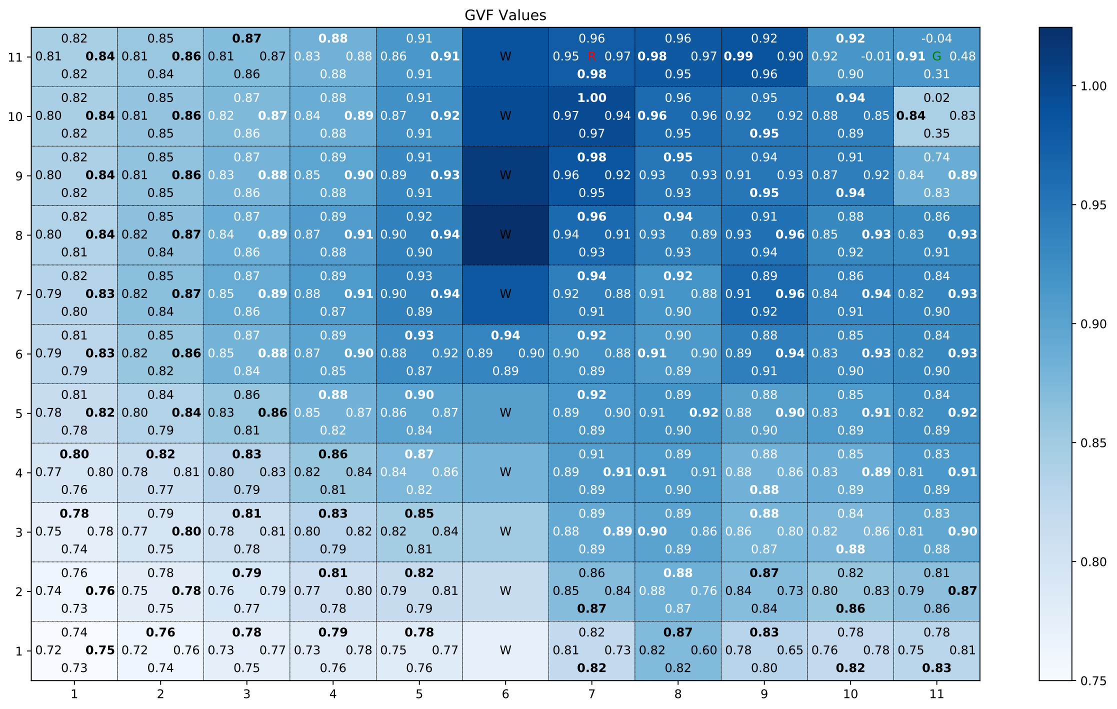
Conclusion and Future Work

From the experiments, Directed EZ greedy seems to learn much much faster because of a better exploration strategy along with improved representation learning. Intuitively as well, this makes sense as options generated from the GVFs take the agent into states which are useful for optimizing towards the main goal. However, since this architecture is very flexible as we only require a scalar reward to learn the GVFs.

The next steps would be to try it on exploratory demanding tasks. Additionally, it would be useful to see how it compares to traditional curiosity driven exploratory based methods. Additionally, DEZ-greedy can be combined with any intrinsic motivation based exploration strategy, as well, if they add exploratory reward functions to the main reward. This along with goal-driven GVFs would further improve exploration for complicated environments. It can also be used for transfer learning in procedurally generated environments as the GVFs learned can help in generating useful options for newer tasks as well.



(a) EZ-DQN+GVF



(b) DEZ-DQN+GVF

Figure 5: Off-policy divergence with generic GVF algorithms for 11x11 SubGoal Two Rooms. The table shows Q values of the GVF that gets a reward of +1 on collecting the red dot. The values are much more bounded for DEZ-greedy exploration.

Table 1: Hyper-Parameters

Environment	Algorithm	Learning Parameters	Model Parameters
Two Rooms	DQN DQN+GVF EZDQN EZDQN+GVF DEZDQN+GVF	runs = 10 episodes = 2000 batch = 64 γ = 0.99 α = 0.001 ϵ_{start} = 1.0 ϵ_{stop} = 0.001 ϵ_{decay} = 1e-3**(1/episodes) Persistence, Z_{max} = 30 GVFs = 1	Architecture = [16 units, 16 units] Replay Buffer Size = 10000 Target Network Update = 100 steps
Sub Goal Two Rooms	DQN DQN+GVF EZDQN EZDQN+GVF DEZDQN+GVF	runs = 10 episodes = 2000 batch = 64 γ = 0.99 α = 0.001 ϵ_{start} = 1.0 ϵ_{stop} = 0.001 ϵ_{decay} = 1e-3**(1/episodes) Persistence, Z_{max} = 30 GVFs = 2	Architecture = [16 units, 16 units] Replay Buffer Size = 10000 Target Network Update = 100 steps

References

Badia, A. P.; Piot, B.; Kapturowski, S.; Sprechmann, P.; Vitvitskyi, A.; Guo, D.; and Blundell, C. 2020a. Agent57: Outperforming the Atari Human Benchmark. arXiv:2003.13350.

Badia, A. P.; Sprechmann, P.; Vitvitskyi, A.; Guo, D.; Piot, B.; Kapturowski, S.; Tieleman, O.; Arjovsky, M.; Pritzel, A.; Bolt, A.; and Blundell, C. 2020b. Never Give Up: Learning Directed Exploration Strategies. arXiv:2002.06038.

Dabney, W.; Ostrovski, G.; and Barreto, A. 2020. Temporally-Extended ϵ-Greedy Exploration. *CoRR*, abs/2006.01782.

Ecoffet, A.; Huizinga, J.; Lehman, J.; Stanley, K. O.; and Clune, J. 2019. Go-Explore: a New Approach for Hard-Exploration Problems. *CoRR*, abs/1901.10995.

Fortunato, M.; Azar, M. G.; Piot, B.; Menick, J.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; Pietquin, O.; Blundell, C.; and Legg, S. 2017. Noisy Networks for Exploration. *CoRR*, abs/1706.10295.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. Reinforcement Learning with Unsupervised Auxiliary Tasks. arXiv:1611.05397.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602.

Munos, R.; Stepleton, T.; Harutyunyan, A.; and Bellemare, M. G. 2016. Safe and Efficient Off-Policy Reinforcement Learning. arXiv:1606.02647.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven Exploration by Self-supervised Prediction. *CoRR*, abs/1705.05363.

Péré, A.; Forestier, S.; Sigaud, O.; and Oudeyer, P. 2018. Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration. *CoRR*, abs/1803.00781.

Revaud, J.; Weinzaepfel, P.; Souza, C. D.; Pion, N.; Csurka, G.; Cabon, Y.; and Humenberger, M. 2019. R2D2: Repeatable and Reliable Detector and Descriptor. arXiv:1906.06195.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587): 484–489.

Sutton, R. S.; Modayil, J.; Delp, M.; Degris, T.; Pilarski, P. M.; White, A.; and Precup, D. 2011. 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*, AAMAS ’11, 761–768. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0982657161.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1): 181–211.

Tang, H.; Houthooft, R.; Foote, D.; Stooke, A.; Chen, X.; Duan, Y.; Schulman, J.; Turck, F. D.; and Abbeel, P. 2017. Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. arXiv:1611.04717.

Veeriah, V.; Hessel, M.; Xu, Z.; Lewis, R.; Rajendran, J.; Oh, J.; van Hasselt, H.; Silver, D.; and Singh, S. 2019. Discovery of Useful Questions as Auxiliary Tasks. arXiv:1909.04607.