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

Durgesh Kalwar
TCS Research
Mumbai, India
durgesh.kalwar@tcs.com

Omkar Shelke
TCS Research
Mumbai, India
shelke.omkar@tcs.com

Somjit Nath
TCS Research
Mumbai, India
somjit.nath@tcs.com

Hardik Meisheri
TCS Research
Mumbai, India
hardik.meisheri@tcs.com

Harshad Khadilkar
TCS Research
Mumbai, India
harshad.khadilkar@tcs.com

ABSTRACT

Improving sample efficiency is a key challenge in reinforcement learning, especially in environments with large state spaces and sparse rewards. In literature, this is resolved either through the use of auxiliary tasks (subgoals) or through clever exploration strategies. Exploration methods have been used to sample better trajectories in large environments while auxiliary tasks have been incorporated where the reward is sparse. However, few studies have attempted to tackle both large scale and reward sparsity at the same time. This paper explores the idea of combining exploration with auxiliary task learning using General Value Functions (GVFs) and a directed exploration strategy. We present a way to learn value functions which can be used to sample actions and provide directed exploration. Experiments on navigation tasks with varying grid sizes demonstrate the performance advantages over several competitive baselines.

KEYWORDS

General value functions; Directed exploration; Navigation task

1 INTRODUCTION

Reinforcement Learning (RL) has shown great promise in recent years for solving sequential decision making tasks. Particularly in the field of games, Reinforcement Learning algorithms have managed to achieve superhuman levels of performance[9] [14]. A recurring theme in the design of RL algorithms is the exploration vs exploitation tradeoff, particularly in hard exploration environments. It is a common practice to use ϵ-greedy strategies for exploration. However, for hard exploration environments this strategy is unable to learn efficiently due to the low probability of generating sensible trajectories with purely random exploration. This property has

been observed in environments such as the game Montezuma's Revenge in Atari [9].

The main goal in RL tasks is to obtain an optimal policy that maximises the expected discounted return (with discount factor, $\gamma \in [0, 1]$). The optimal policy, π^* can be defined as:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \ \mathbb{E}_\pi \big[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t) \big]$$

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 [6], Count-based Exploration [17], Curiosity Driven Exploration [11], unsupervised learning of goal space for intrinsically motivated exploration [12], Go-explore [5] 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 [2] paper, which was included in the Agent 57 [1] 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) [13] with Retrace (λ) [10] 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, [4] came up with a temporally extended version of the ϵ-greedy exploration strategy that can handle complicated environments as well.

As explained in [4], the various other exploration strategies developed keeping difficult environments in mind tend to over-fit to those challenging environments and do not work as well in environments requiring lesser exploration. The authors further explain how temporally extended epsilon greedy is a sound strategy that can work well in a variety of environments. In this paper, we aim to demonstrate the following contributions:
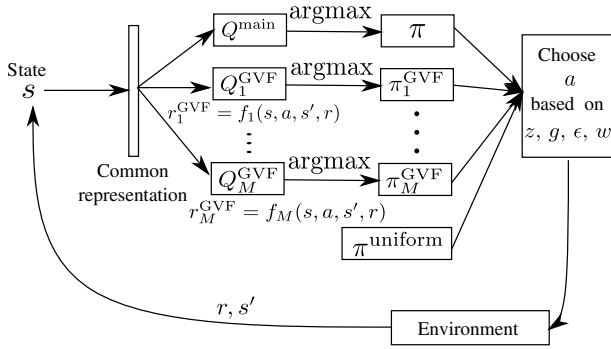
Figure 1: High-level architecture of the DEZ-exploration policy with M GVFs, one main policy, and one uniform policy. The reward for training each GVF is computed using handcrafted functions f. The logic for choosing action a is described in Algorithm 1.

(1) We present an exploration strategy designed for large state spaces with sparse rewards. This is achieved by defining auxiliary tasks with the help of General Value Functions (GVFs) to perform directed exploration, thereby further improving state space coverage during exploration. Our strategy improves on temporally extended ϵ-greedy exploration with sub-policies from the General Value Functions to perform better exploration.

(2) Extensive experiments over two sets of navigation environments, which provide empirical evidence about superior performance and robustness to hyperparameters such as the ϵ schedule.

2 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 time, it would explore the entirety of the state space as it would be taking random actions infinite times provided we 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 [4] fixes a random action and keeps taking the action for a certain number of time steps, called persistence (Z) of the random action. Instead of taking one random step with probability ϵ, it executes a sequence of the same random actions. The persistence Z is not a constant value throughout the exploration process however it is sampled from a uniform distribution between 0 (no exploration) and Z_{max} (a hyper-parameter specific to 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 instead, we show that sampling actions from the greedy policy of learned General Value Functions (GVFs) [16] K times yields better-directed exploration. This can help to reach parts of state-space that could be novel or would help in solving the main goal. In this paper, we show that having GVFs which are trained over auxiliary tasks provides an efficient way to explore as

compared to count-based learning (curiosity-driven learning) or reward shaping.

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) [16] estimate a user-specified environment property, optimizing over the user-specified 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). The 'question' asked of a GVF is the estimate of the discounted return for the specified cumulants. For each question, 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 [16], the authors show the usefulness of answering predictive questions about the environment as well as the capability of learning 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. The tasks may be defined using handcrafted cumulants [7] or they may themselves be learnt on their own [18]. In this paper, we define the GVF reward functions and termination functions by hand, and learn the GVF values off-policy i.e. they are learned with respect to their own greedy policies. Thus, maximising for Q^{GVF} generates a different sub-task oriented policy which can be viewed as an subroutine, to be used for a short persistence period. Thus, by following the policy for a short period we can explore directed by a subpolicy.

3 PROPOSED DIRECTED EZ-GREEDY STRATEGY

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 for temporally extended ϵ-greedy algorithm. With probability ϵ 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 values. In all our experiments, we include the option of repeating a random action as one additional option as well. This is added to make sure that theoretically, it is still possible to visit all states in the state space and we have asymptotic full state space coverage.

There are two main advantages of this approach:

**Algorithm 1:** DEZ-Greedy exploration strategy

Function DEZGreedy(ϵ, Z_{max}):
    Countdown timer $z \leftarrow 0$
    Uniformly sampled random action $w \leftarrow -1$
    Selected GVF index $g \leftarrow 0$
    while *True* **do**
        Observe state s
        if $z == 0$ **then**
            // Timer ran out: choose controlling policy
            if $random() < \epsilon$ **then**
                // Explore
                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
                    $a \leftarrow argmax(Q_g^{GVF})$
                end
            else
                // Exploit
                $a \leftarrow argmax(Q^{Main})$
            end
        else
            // Continue previous exploration policy
            if $g == 0$ **then**
                $a \leftarrow w$
            else
                $a \leftarrow argmax(Q_g^{GVF})$
            end
            $z \leftarrow z - 1$
        end
        Take action a
    end

(1) GVFs have been generally used to give predictive knowledge of the environment and this directly helps in learning good representations. In Figure 1, we also employ a similar structure whereby the GVFs not only influence the exploration policies but also help in learning better representations.

(2) 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).

4 ENVIRONMENTS

Our experiments were tested in the gridworld setting with hand-crafted cumulants. We mainly considered following 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 (the red dot is not relevant in this environment). This seems like an easy task, but for larger dimensions it becomes
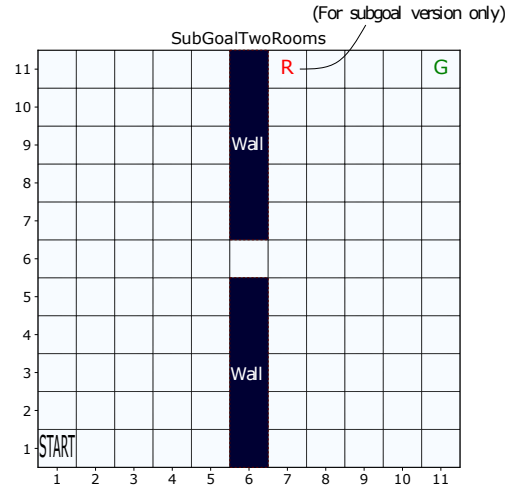
Figure 2: Two Rooms & SubGoal Two Rooms Environment. The red dot (SubGoal) is represented by R, and the main goal (for both environments) is the green dot shown here as G.

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 terminal reward on reaching the green dot as follows:

$$Reward = 1.0 - 0.9 \times (t/t_{max})$$

where t is current time-step and t_{max} is maximum time-step in a episode. Episodes terminates after M=maximum time step of grid-dim×10. 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 goal position is as follows

$$Reward = \begin{cases} 1.0 - 0.75 \times (t/t_{max}), & \text{if visited red dot} \\ 0.2 - 0.1 \times (t/t_{max}), & \text{else} \end{cases}$$

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 more GVF in addition to the one used in Two Rooms, which receives a cumulant of +1 if the agent visits the red dot.

- **Doorkey Environment:** This is popular environment for benchmarking RL algorithms [3], with different tasks. For our experimentations we chose Doorkey environment with 6×6 and 8×8 grid-size. Board configuration is very similar to Two Rooms environment however, position of corridor and initial position of agent are random. In addition, to enter another room, agent needs to pickup a key, unlock and open the door. Additionally, the actions for going left and right are a bit different than just going left or right. The agent has

to first turn to the direction it needs to move in and then go forward. This can be really problematic for EZ-Greedy methods as action repetition would get nowhere and the agent would basically keep on rotating in the same state. Thus, to make the comparison fair, we included options for going left and right for these environments. Additionally, for this environment, we used 2 GVFs. The first GVF gets a cumulant of +1 for picking up the key and the next GVF, gets a cumulant of +1 for unlocking the door.

5 RESULTS

In this section, we will compare our approach with ϵ-greedy strategy that is widely used as well as the temporally extended ϵ-greedy strategy [4]. The detailed hyper-parameters for reproduction of the results are listed in Appendix B of supplementary material[1].
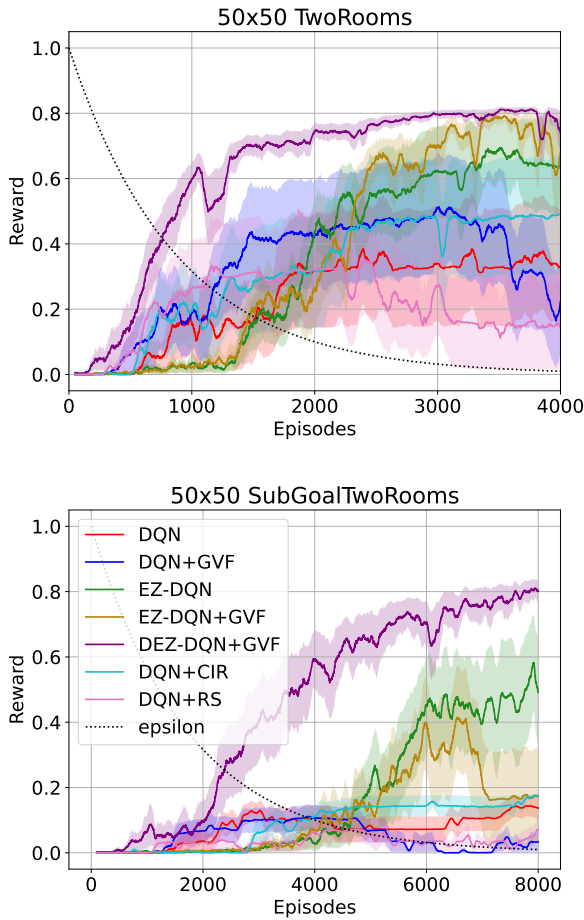
Figure 3: Different Algorithms compared on TwoRooms 50x50 ($Z_{max} = 10$) and SubGoal TwoRooms 50x50 ($Z_{max} = 30$) Environment, shaded region denotes standard deviation across 5 seeds.

---

[1]Supplementary material at https://arxiv.org/abs/2203.00874

Algorithms compared: For our experiments, we compare 7 different algorithms.

- The RL algorithm used for our experiments is the DQN algorithm [9], 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. Details about the variants of DQN used for each of the environments are presented in Appendix B (supplementary).

- EZ-DQN is the temporally extended ϵ-greedy version of DQN with EZ-Greedy exploration. The version with repetitive actions is used Two Rooms and SubGoal Two Rooms Environments. But for DoorKey, we used options for going left, right and up instead of repetitive actions. EZ-DQN with repetitive actions for DoorKey is denoted by EZ-DQN(Vanilla).

- 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 sampling actions for directed exploration

- DQN+CIR Count-based Intrinsic Reward policy is count based learning method in which additional intrinsic signal is used to encourage exploration [15]. The policy is thus trained with the reward composed of two terms $r_t = r_e + \eta \times r_i$. where η is a hyper-parameter and $r_i = 1/\sqrt{N(S_t)}$. Authors in [15] propose $1/\sqrt{N(S_t, A_t)}$, we have limited ourselves to counting unique states. This can serve as a proxy for curiosity driven learning.

- DQN+RS Reward shaping [8] is policy in which positive rewards are given when subgoals are reached, this is an alternative method to incorporate domain knowledge.

For our paper, we did not compare any exploratory methods apart from [4], since this reference is shown to beat other state-of-the-art algorithms. Thus we only used their baseline in addition to epsilon greedy, which is the widely used exploration strategy. We ideally want to ask, what would be a good exploratory strategy that can work well across a bunch of different kinds of environments (both simple and challenging).

5.1 Two Rooms & SubGoal Two Rooms

The Figure 3 portrays the learning curves of the algorithms across 5 seeds over 4000 episodes for TwoRooms and 8000 episodes for SubGoal TwoRooms environment. The shaded regions denotes the standard deviation across 5 seeds for all the training curves.

It is evident that EZ-Greedy and in particular taking repeated actions helps in exploring the state space and there is a marked difference between algorithms that use persistence and normal ϵ-greedy. More so in SubGoal Two Rooms environment where ϵ-greedy algorithms have converged to sub-optimal values. Adding directed exploration with the use of GVFs makes the RL agent outperform EZ-Greedy because of the exploration of useful state spaces as the GVFs are designed to stimulate goal-driven exploratory behaviour.
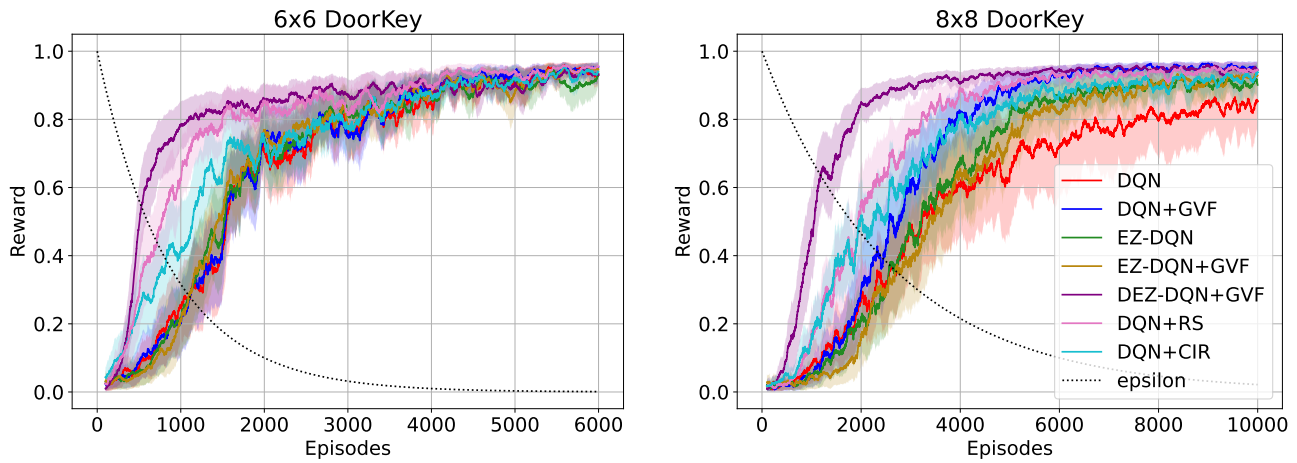
Figure 4: Training plot for DoorKey environment ($Z_{max} = 3$), shaded region denotes standard deviation across 5 seeds.
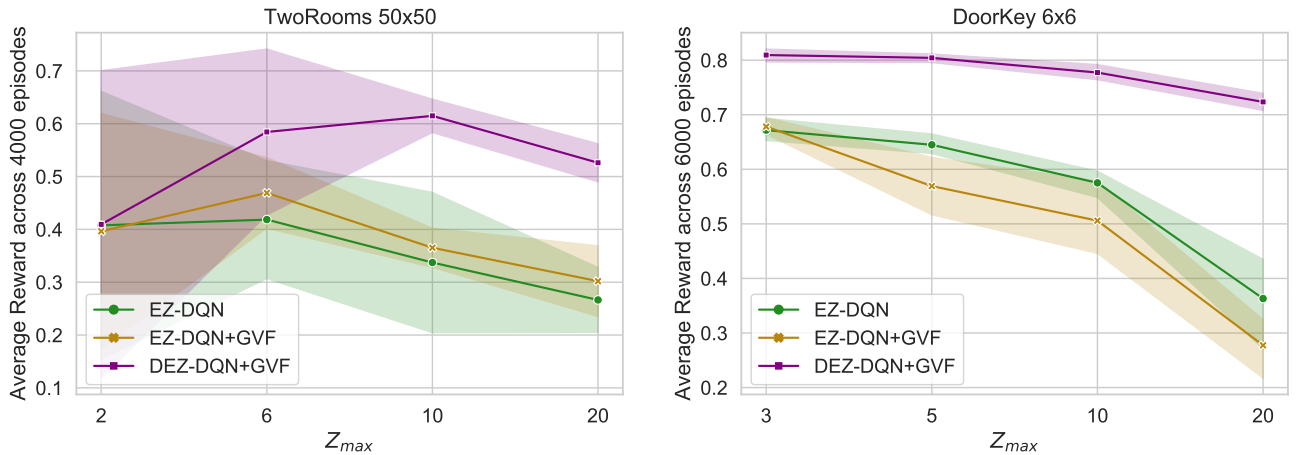

Figure 5: Parameter Sensitivity of EZ-Greedy and DEZ-Greedy strategies with respect to the persistence, Z_{max} in (a) TwoRooms 50x50 and (b) Doorkey 6x6 Environment. Shaded region denotes 95% confidence interval across 5 random seeds.

Another interesting thing worth noting is that even adding GVFs without using any exploratory strategy (DQN+GVF blue) helps in better performance in Two Rooms environment compared to its counterpart (DQN red). This is likely due to GVFs learning useful representations which can improve performance. However, there is performance degradation in SubGoal Two Rooms and we believe this is due to policies converging to sub optimal values and inherent sample inefficiency. A similar pattern is observed for EZ-Greedy with GVFs there is an improvement in performance in the Two Rooms environment, while slight drop in SubGoal Two rooms environment. In addition, DQN+CIR is able to outperform vanilla DQN due to better exploration characteristics in Two Rooms however, the count-based exploration strategy is still not sufficient to elevate the DQN from suboptimal point in SubGoal Two rooms environment.

5.2 DoorKey

In Figure 4 we present training results for 2 grid sizes; 6×6 and 8×8 of the Doorkey environment from Gym MiniGrid [3]. For both the gridsize we can observe DEZ-DQN+GVF converging early compared to other algorithms, the difference is more pronounced in higher grid size. For 8×8 size, we see higher variance across random seeds for all the baselines whereas DEZ-DQN+GVF has relatively low variance and shows a similar trend to 6×6 providing evidence of robustness across grid sizes, as observed earlier. It is worth noting that having reward shaping (DQN+RS) which is a proxy for introducing additional information into the learning does help the vanilla DQN and shows improvement over other baselines. Similarly, DQN+CIR does get a steep rise in rewards thanks to reaching novel states earlier than other baselines however it does taper off as the training progress.
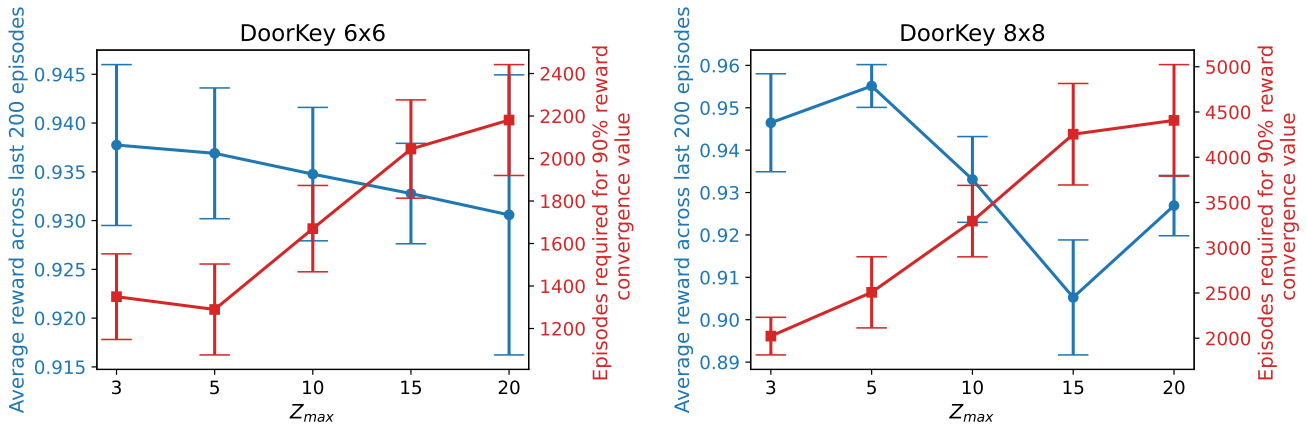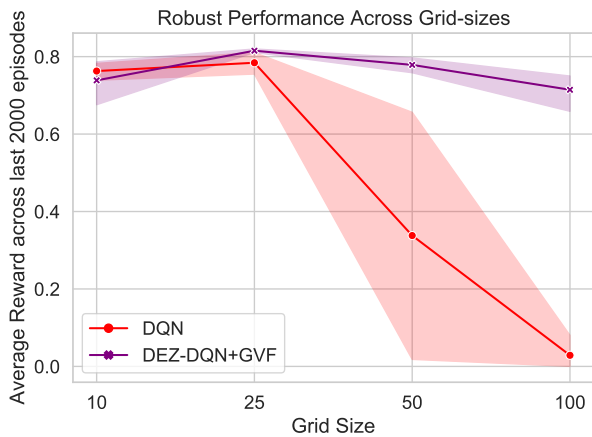
Figure 6: DEZ-Greedy performance for different persistence values on Doorkey environment. The Y-axis on the left represents the average reward across the last 200 games while the axis on the left displays the number of episodes DEZ-DQN+GVF took to reach 90% of the final convergence value of the reward. Error bar denotes the standard deviation across 5 random seeds.

It is also interesting to note there is little or no improvement in the performance of EZ-DQN+GVF over EZ-DQN for 8×8 DoorKey which is in stark contrast to Two Rooms and SubGoal Two Rooms environments. We believe this is due to the off-policy divergence that can happen when the greedy policy with respect to each of the GVFs is very different from the behaviour policy of the main RL agent (also amplified by k repeated actions). This phenomenon discourages the use of GVFs for representation learning particularly for Deep RL methods because the divergence in GVFs values results in degradation in performance. However, our method periodically takes actions from the GVFs, it helps in avoiding diverging values.


Figure 7: Performance comparison of DQN and DEZ-DQN+GVF for various grid sizes of TwoRooms Environment

Sensitivity with respect to Persistence Z_{max}: 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 5, we plot the sensitivity of the algorithms across different persistence values for a) 50×50 Two Rooms environment and b) door key environment with 6×6 grid size. We can observe that there is an optimal value of persistence which provides the best performance across the algorithms, while lower and higher values result in degradation of performance. Lower values would essentially lead to ϵ greedy exploration whereas high values would result in agent sticking to boundaries in case of random sampling. However, DEZ is much more robust to these kinds of situations and provides much more reliable performance.

To further understand the effect of Z_{max} on convergence characteristics, Figure 6 portrays the performance of DEZ-DQN+GVF on 6×6 and 8×8 DoorKey with respect to increasing Z_{max}. Two Y-axis represent the average reward across the last 200 episodes (blue) and the number of episodes taken to reach the 90% of convergence value of the reward (red). The spread of values across the left Y-axis (blue) is very minimal for 6×6, and within a reasonable range for 8×8. As expected only change noticeable change happens in the red line, indicating higher Z_{max} values take longer to reach convergence value due to the inherent smaller grid size of the environment. With a high value for Z_{max} in a smaller grid size EZ-DQN algorithm leads the agent to the edge of the grid sizes, whereas for DEZ-DQN+GVFit quickly achieves the GVF goal. In both of these algorithms actions are wasted and hence slower convergence with higher Z_{max} however, with DEZ-DQN+GVFfinal performance is still not degraded significantly as the agent is closer to achieving the goal than before.

Sensitivity with respect to grid size: One aspect that we wanted to target is better performance for both simple and hard domains. In Figure 7 we plot the mean average reward for ϵ-greedy and DEZ-Greedy method. With smaller grid size, ϵ-greedy does really well but it fails to work for higher grid sizes. However, DEZ-greedy not only works for the higher grid sizes, but also has comparable performance to ϵ-greedy in smaller domains as well. This is an interesting result because even it shows how this strategy is robust to the hardness of the environment with respect to exploration. Thus

DEZ-Greedy can be used universally for all types of environments. Here for DEZ-Greedy method for grid-sizes 10, 25, 50 and 100 we use 5, 5, 10 and 20 Z_{max} values respectively.
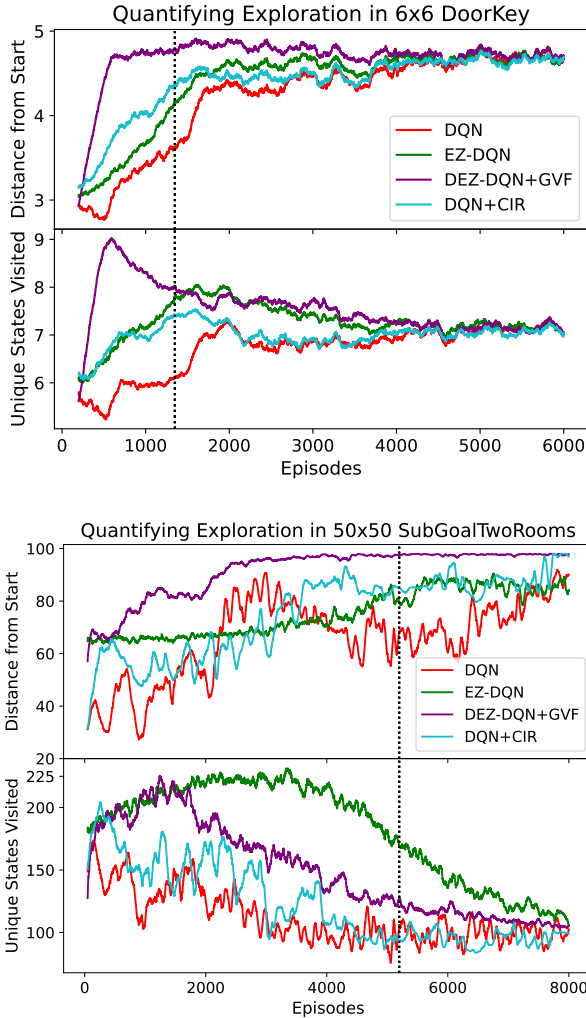
Figure 8: This plot tries to quantify exploration by comparing distance and unique states visited by normal epsilon greedy and directed epsilon greedy method. The graph is plotted with aggregating values for 100 episodes. a) 6x6 DoorKey b) 50x50 SubGoal TwoRooms

Quantifying Exploration: Having a higher reward is one way of ascertaining that the agent has explored important parts of the state space faster and as a result is able to converge much faster. However, a direct comparison of the exploration strategies can be made by using visitation frequency of unique states. Figure 8 presents a comparison of various exploration strategies directly by plotting the farthest distance from the initial state and the number of unique states visited. Distance from the initial state is also quantification of exploration for this environment as the distance would increase
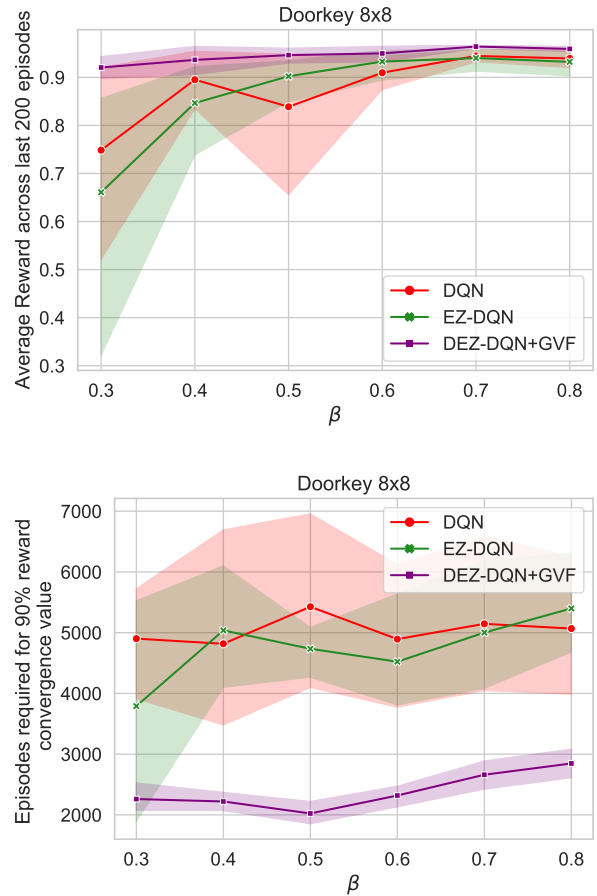
Figure 9: Performance of different algorithm for different β values. This plot tries to quantify the robustness of different algorithm with respect to rate of epsilon decay for exploration. Shaded region denotes 95% confidence interval across 5 random seeds

when the agent can unlock the door and visit the next room. Vertical line indicates the episode when DEZ-DQN+GVF reached 90% of its convergent value. We can observe that directional exploration is helping DEZ to not only explore farther from the initial state but also reach unique states quickly. In addition, number of unique states quickly goes down as it reaches it 90% convergence value providing enough evidence that it has found its optimal policy way early compared to baselines.

To further understand the robustness of DEZ with respect to exploration, we define epsilon decay as

$$\epsilon_{decay} = (0.01)^{\frac{1}{episodes \times \beta}}$$

where $\beta \in (0, 1]$ is a hyper-parameter. So for small β value the rate of epsilon decay will be fast and the agent will have less time for exploration. Figure 9 a) shows the reward achieved while training with different β epsilon schedule, we can observe that DEZ-DQN+GVF has little to no variance even during less value of β and minute drop

in convergence value. This suggests that it is much more robust to the exploration schedule and achieves better results even with less exploration. This hypothesis is further reinforced by results in Figure 9 b), where we can observe that number of episodes required to reach 90% of the convergence value is much lower compared to baselines and increases marginally with respect to β.

Algorithm Overhead: In general, online Reinforcement Learning does not suffer from the problem of computational complexity, but more from sample inefficiency. Specially in real world settings when the number of interactions are limited, DEZ-Greedy can be really useful, because it requires much fewer samples for better performance. In terms of compute, yes, it does add compute because it involves training General Value Functions, in addition to the main RL network. However we note that this is parallelizable during training; and it makes no effect on post-training inference since we use only the main policy. Training time of DEZ and EZ for 50x50 two-rooms env for 4000 episodes is 1hr 7min and 55 min respectively. The hardware used for the experiments consist of an Nvidia A100 GPU with 14 vCPUs.

6 CONCLUSION AND FUTURE WORK

Exploration is one of the critical aspects for good performance for Reinforcement Learning, especially in games. EZ-Greedy [4] 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. From the experiments, DEZ-Greedy seems to learn much faster because of a better exploration strategy along with improved representation learning. Intuitively as well, this makes sense as actions generated from the GVFs take the agent into states which are useful for optimizing towards the main goal. Additionally, this architecture is very flexible as we only require a scalar reward to learn the GVFs.

The next steps would be to try to adapt it to environments which can learn the cumulants automatically from the environments [18]. It can also be used for transfer learning in procedurally generated environments as the GVFs learned can help in generating useful directed exploration for newer tasks as well.

REFERENCES

[1] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. 2020. Agent57: Outperforming the Atari Human Benchmark. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 507–517. http://proceedings.mlr.press/v119/badia20a.html

[2] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. 2020. Never Give Up: Learning Directed Exploration Strategies. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=Sye57xStvB

[3] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2018. Minimalistic Gridworld Environment for OpenAI Gym. https://github.com/maximecb/gym-minigrid.

[4] Will Dabney, Georg Ostrovski, and André Barreto. 2021. Temporally-Extended ϵ-Greedy Exploration. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=ONBPHFZ7zG4

[5] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2019. Go-Explore: a New Approach for Hard-Exploration Problems. *CoRR* abs/1901.10995 (2019). arXiv:1901.10995 http://arxiv.org/abs/1901.10995

[6] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. 2018. Noisy Networks For Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rywHCPkAW

[7] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. 2017. Reinforcement Learning with Unsupervised Auxiliary Tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJ6yPD5xg

[8] Adam Daniel Laud. 2004. *Theory and application of reward shaping in reinforcement learning*. University of Illinois at Urbana-Champaign.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[10] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. 2016. Safe and Efficient Off-Policy Reinforcement Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 1046–1054. https://proceedings.neurips.cc/paper/2016/hash/c3992e9a68c5ae12bd18488bc579b30d-Abstract.html

[11] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 2778–2787. http://proceedings.mlr.press/v70/pathak17a.html

[12] Alexandre Péré, Sébastien Forestier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2018. Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=S1DWPP1A-

[13] Jérôme Revaud, César Roberto de Souza, Martin Humenberger, and Philippe Weinzaepfel. 2019. R2D2: Reliable and Repeatable Detector and Descriptor. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 12405–12415. https://proceedings.neurips.cc/paper/2019/hash/3198dfd0aef271d22f7bcddd6f12f5cb-Abstract.html

[14] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529, 7587 (jan 2016), 484–489. https://doi.org/10.1038/nature16961

[15] Alexander L. Strehl and Michael L. Littman. 2008. An analysis of model-based Interval Estimation for Markov Decision Processes. *J. Comput. System Sci.* 74, 8 (2008), 1309–1331. Learning Theory 2005.

[16] Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. 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* (Taipei, Taiwan) *(AAMAS '11)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 761–768.

[17] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2017. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 2753–2762. https://proceedings.neurips.cc/paper/2017/hash/3a20f62a0af1aa152670bab3c602feed-Abstract.html

[18] Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Janarthanan Rajendran, Richard L. Lewis, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. 2019. Discovery of Useful Questions as Auxiliary Tasks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 9306–9317. https://proceedings.neurips.cc/paper/2019/hash/10ff0b5e85e5b85cc3095d431d8c08b4-Abstract.html