IMPROVING GENERALIZATION OF REINFORCEMENT LEARNING USING A BILINEAR POLICY NETWORK

Fen Fang[1], *Wenyu Liang*[1], *Yan Wu*[1], *Qianli Xu*[1] *and Joo-Hwee Lim*[1,2]

[1]Institute for Infocomm Research, A*STAR, Singapore

[2]School of Computer Science and Engineering, NTU, Singapore

ABSTRACT

In deep reinforcement learning (DRL), the agent is usually trained on seen environments by optimizing a policy network. However, it is difficult to be generalized to unseen environments properly, even when the environmental variations are insignificant. This is partly because the policy network cannot effectively learn the representation of visual difference that is subtle among highly similar states in the environments. Because a bilinear structured model containing two feature extractors allows pairwise feature interactions in a translationally invariant manner which makes it particularly useful for subtle difference recognition among highly similar states, in this work, a bilinear policy network is employed to enhance representation learning, and thus to improve generalization of the DRL. The proposed bilinear policy network is tested on various DRL task, including a control task on path planning for active object detection, and Grid World, an AI game task. The test results show that the generalization of DRL can be improved by the proposed network.

Index Terms— Deep Reinforcement Learning, Bilinear Policy Network, Generalization.

1. INTRODUCTION

In deep reinforcement learning (DRL), an agent is trained to learn the optimal actions corresponding to varying states in the dynamic environment. For an environment, especially a complex one that contains a large state space, some states are highly similar to one another. In such environment, the policy network has to extract distinctive features from these states independently by analyzing tiny differences among them. Otherwise, the policy network is prone to make identical action decisions on these similar states [1]. This leads to two undesirable outcomes: (i) it will be very difficult for the agent being trained to converge within reasonable time; (ii) the policy cannot generalize to environments different from those the agent was trained on [2]. For example, in a simple obstacle avoidance task, an agent (white block) shown in Fig. 1 learns from pixel images to jump to avoid an obstacle (grey square).

This research is supported by the Agency for Science, Technology and Research, under the AME Programmatic Funding Scheme (Project#A18A2b0046).

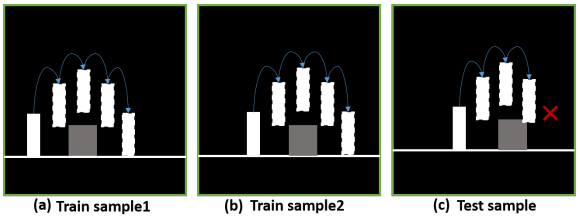


Fig. 1. Obstacle avoidance task training and testing samples. An agent (white block) strives to jump over the obstacle (gray square) with different distances from the agent by taking optimal actions (blue trajectories). The task fails if the agent bumps into the obstacle.

The agent is trained via an RL algorithm on a lot of tasks in seen environments with varying obstacle locations before it is hoped to successfully jump on the test tasks in unseen environments with subtle obstacle location differences. The challenge of the task is to generalize the action decision policy based on obstacle locations and floor heights in seen environments to the tasks in unseen environments within reasonable training episodes. Fig.1 (a) and (b) show two sample tasks in training with the optimal trajectories of the agent shown as blue curved arrows; (c) is one of the test tasks. Since the pixel image of the test task is very similar to that of the training sample in (b), the agent is prone to take the same action sequence to jump over the obstacle in the test task. However, due to the subtle visual differences in floor height and obstacle size in the images, the agent will bump onto the obstacle when taking the 4th action which fails the task. The example illustrates the gap of current DRL methods in dealing with subtle visual differences of train-and-test environment. A stronger feature representation learning scheme is critically needed.

Bilinear construct was initially employed in support vector machine (SVM) classifier for human detection and action classification in video sequences [3]. It has also been used in neural networks (e.g., bilinear convolutional neural network (BCNN)) [4, 5, 6, 7, 8] with performance on public datasets verifying that it works effectively and efficiently on fine-grained object recognition task. The main advantage lies in its ability to generalize several widely-used texture descriptors (e.g., Fisher Vector [9], SIFT features [10]). In addition, it can be trained end-to-end. Inspired by the success of BCNN in fine-grained object recognition, this study proposes a novel

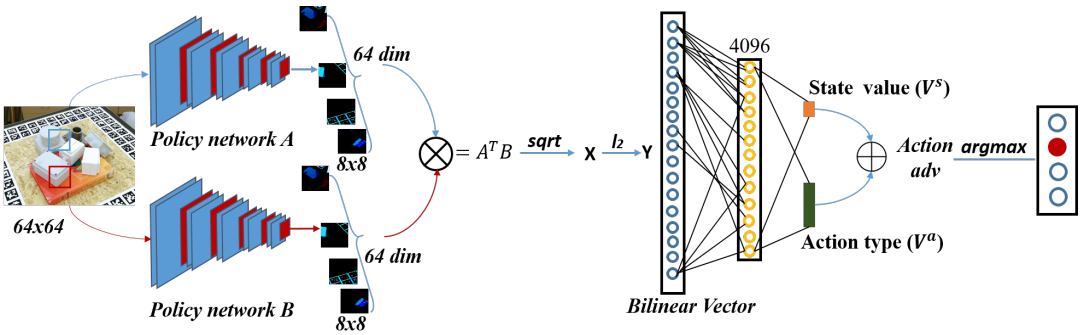


Fig. 2. The architecture of our proposed method.

method to enhance the generalization of DRL policy that is learnt on seen environments and applied in unseen environments by using bilinear policy network. Bilinear structure has been utilized in RL for better generalization in [11], but there is a key difference. In [11], Bilinear structure is employed to improve generalization of RL by Q-value-shaping, in our method, it is used to improve RL generalization by policy-shaping. The framework of the proposed method is shown in Fig. 2. In the proposed method, each image goes through both the Policy network A and the Policy network B, of which outputs are combined at each location of the image using *outer product*. Bilinear feature representation is thus obtained by taking *average pooling*, *square root*, and *L2 regularization*. The extracted feature is then linearly converted to a 1-D vector-*state value*, and an n-D vector-*action type*, where n is the action space. The two vectors are added together to compute the action advantage vector over average. Finally, the *argmax* operation is applied on the action advantage vector to make the optimal action decision on the current state (i.e., the input image). The bilinear structured model allows pairwise feature interactions in a translationally invariant manner which makes it particularly useful for subtle difference recognition among highly similar states. Thus, better generalization of the learnt policy from seen environments to unseen environments can be achieved by applying bilinear policy network in DRL.

2. OUR METHOD

In general, key components of a standard DRL are states, reward, environment, and policy network. In this work, all these components except the policy network are task-dependent, details of which will be presented in the experiment section. In this section, we focus on the proposed bilinear policy network and the specific RL algorithm illustration.

2.1. Bilinear Policy Network

The purpose of DRL is to learn how to maximize the reward by training a policy network over varying states from an environment. The policy network is usually a linear model:

$$f_w(x) = w^T x = \sum_{i=1}^{n} w_i x_i, \tag{1}$$

where x is the feature vector and w is a weight vector to specify how important the feature is to the model to predict the action at the current state, and f is the feature function. However, linear model has many problems, especially in visual recognition tasks, where the feature is more naturally described in the form of tensor or matrix rather than a vector [3]. Alternatively, a bilinear model can be used to extract feature in vision task, which is defined as

$$B(x, y) = x^T W y = \sum_{i,j=1}^{n} w_{ij} x_i y_j. \tag{2}$$

Basically, the bilinear combination of two stream of a CNN model at a location l of an image I is computed by

$$B(f_A, f_B, l, I) = f_A(l, I)^T f_B(l, I), \tag{3}$$

where f_A and f_B are feature functions that have the same dimension. The bilinear combination is the outer product between the outputs of features f_A and f_B. The global image representation ($\phi(I)$) of bilinear combination features over the entire image using sum pooling can be computed by

$$\phi(I) = \sum_{l \in \Gamma} B(f_A, f_B, l, I). \tag{4}$$

Assuming that f_A and f_B are of dimensions $D \times M$ and $D \times N$, respectively, $\phi(I)$ is of dimension $M \times N$, and will be reshaped to dimension $MN \times 1$. The bilinear combined feature has some desirable characteristics: (i) it is orderless because the sum pooling operation ignores position information, and (ii) it is a generic descriptor of an image, and hence a classifier (softmax) can be applied on it. To improve the performance for classification, suggested by [9], the signed square root followed by l_2 normalization on x=$\phi(I)$ is added before softmax function:

$$\begin{cases} x_{sr} = sign(x)\sqrt{|x|} \\ x_{nor} = x_{sr} / \|x_{sr}\|_2 \end{cases}. \tag{5}$$

Then, to predict the probability for class c out of C classes, a softmax activation function is computed by

$$\text{softmax}(w_c^T x_{nor}) = \frac{e^{w_c^T x_{nor}}}{\sum_{j=1}^{C} w_c^T x_{nor}}, \tag{6}$$

where w_c is a learnable vector of dimension $M \times N$ that predicts the probability of the vector x_{nor} belonging to the c-th.

Based on the above description, the key idea of the bilinear model is to explicitly consider feature interactions by applying outer product on outputs of two steams of the policy

network, similarly to the polynomial kernel in SVMs [12]. This makes bilinear model particularly useful for identifying subtle differences, such as fine-grained image recognition [13, 14]. Similarly, it is also helpful for tiny visual difference localization among highly similar states in DRL. The detailed description of shared convolutional neural network architecture in the bilinear policy network is shown in Fig. 3.

2.2. Reinforcement Learning Algorithm

In supervised learning, model training can be started once CNN model is constructed because the loss gradient is computed by minimizing error between model output and ground truth pairs. RL does not require preexisting ground truth. Instead, it learns a series of actions by exploring an environment through trial and error [15, 16]. Thus, RL needs to define an algorithm to update the loss gradient for the optimization goal. The RL problem is commonly formulated as a Markov Decision Process (MDP), defined by a tuple (S, A, P, R, γ, π), where A is a action set and S is a state set. In the MDP, there are two functions: a state transition function ($P : S \times A \rightarrow S$) and a reward function ($R : S \times A \rightarrow \mathbb{R}$). The goal of RL is to maximize reward (computed via reward function) by optimizing action selection policy π (through state transition function) for a given task learning. Normally, different RL algorithms have different ways of reaching this goal. In this work, the base RL algorithm is deep Q-learning (DQN) [17] which employs a *state-action value function* defined below to determine optimal policy

$$Q^{\pi}(s_t, a_t) = \mathbb{E}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t, \pi\right\}, \quad (7)$$

where a_t is every action in set A and s_t is each state in set S, r_t is the immediate scalar reward, $\gamma \in [0, 1]$ is the discount factor for balancing current and future reward. The DQN updates every *state-action value* after taking each action a. In turn, the parameter θ of the policy network can be updated according to the following time difference (TD) equation

$$\theta_{t+1} = \theta_t + \alpha[r_t + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1}; \theta^{TD}) - Q(s_t, a_t; \theta_t)] \times \bigtriangledown Q(s_t, a_t; \theta_t), \quad (8)$$

where Q is the *state-action value*. The term α is the update step size. $r_t + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1}; \theta^{TD})$ is the target Q-value, and the term $Q(s_t, a_t; \theta_t)$ is the evaluation Q-value. The mean square error between the two terms is used as loss to be minimized for parameter (θ) optimization. The evaluation Q-value, as shown in Fig.2, is the output of DQN which is the combination of state value and averaged action advantage. It can be computed by the following equation

$$Q(s, a; \theta) = V^s(s; \theta^{bf}, \theta^s) + (V^a(s, a; \theta^{bf}, \theta^a) - \frac{1}{N} \sum_{a' \in A} V^a(s, a'; \theta^f, \theta^a)), \quad (9)$$

where V^s is the 1-D state value vector, V^a is action advantage value, θ^{bf} is the parameters of common part of feature extractor-bilinear network, θ^s and θ^a represent parameters of state value and action, respectively, N is the action space. After training, the optimal action can be selected by applying argmax on the *action advantage* with the learnt policy.

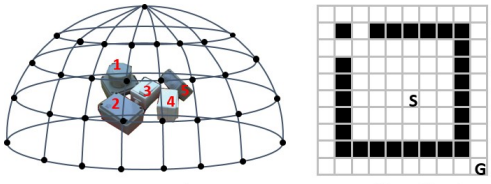
Layer	Kernel Size	Stride	Padding	Nums of filters	Out Shape
Input	—	—	—	—	(64,64,3)
Cov1	5x5	1x1	2x2	16	(64,64,16)
Pool1	2x2	2x2	0	16	(32,32,16)
Cov2	5x5	1x1	2x2	32	(32,32,32)
Pool2	2x2	2x2	0	32	(16,16,32)
Cov3	5x5	1x1	2x2	64	(16,16,64)
Pool3	2x2	2x2	0	64	(8,8,64)
Cov4	3x3	1x1	1x1	64	(8,8,64)
Cov5	3x3	1x1	1x1	64	(8,8,64)

Fig. 3. Description of convolutional neural network (CNN) architecture used in our Bilinear policy network.

3. EXPERIMENTAL SETUP

The proposed method is evaluated on two tasks: (i) Active object detection (AOD) task, which aims to train a robot (agent) to find optimal trajectory to actively identify objects, and (ii) a virtual AI game task, where an agent learns to play in Grid World game [18]. These two tasks are particularly interesting because the adjacent states of them are fairly similar. This presents a tough challenge to the agent when it makes optimal decisions among these states.

In the AOD task as shown in Fig.4 (a), a robot mounted with a pre-trained object detector is trained to actively change its viewpoint (black points) on the grid of semi-spherical surface to obtain useful visual information for the objects identification. A public dataset-TLESS testing set [19] is used in this work. The dataset contains 20 scenes of industrial objects, where objects in each scene have 126 viewpoints (states). To be consistent with [20, 21], benchmarking on the AOD task is implemented on both seen and unseen environments (seen and unseen splits are the same as the ones in [21]). Here, the maximum number of actions per episode is set as 15. Four actions are available for agent to change viewpoint: *left*, *right*, *up* and *down*. If an episode reached desired viewpoint for object identification, it is considered a success; otherwise, it is a failure. The rewards of 1 and -1 are given at the last step of a successful and failed episodes, respectively. Reward is 0 at other steps. For benchmarking, two state-of-the-art methods are employed to speed up AOD of objects: (i) multiple-step action prediction (MAP) method-considering steps size on each action type [22, 21]; (ii) multiple objects prediction (MOP) method-considering AOD of multiple objects at each step [20]. The proposed



(a) AOD task **(b)** Grid World task

Fig. 4. Two tasks used to evaluated our method.

method is implemented on the two methods by replacing the policy network with the bilinear ones. For MAP, the best performance is reported by setting step size space as 4. The same setting is adopted in this work. As MOP explores multiple objects in a single episode, it needs extra information (e.g. oriented bounding box [20]). In the experiment, the benchmarking on MOP method is conducted in two ways: (i) MOP1- only using image as input; (ii) MOP2- using image and extra oriented bounding box as input.

In the Grid World task as shown in Fig.4 (b), the agent is put at the same place (S) at the beginning of each attempt (episode). The goal (G) is at a fixed location; and some fixed walls (black blocks) are distributed in the GridWorld. Once the agent bumps onto a wall, it stops moving. The agent is allowed to take four actions: *left*, *right*, *up* and *down*. The experiments on the task is conducted in two levels of difficulty-state spaces that are of 10×10 (training in 20,000 episodes) and 20×20 (training in 40,000 episodes) grids. When the agent is moving, the reward is 0 at each step unless it encounters one of three scenarios. (i) When the agent bumped onto the wall, the reward is -0.1. (ii) If no further step is allowed for the agent, the reward is -0.1. (iii) If the agent reached the goal, the reward is 1. On this task, the proposed method is benchmarked by the DQN with the linear policy network.

In the agent training process, prioritized experience replay [23] is used to sample a mini-batch of experience for DQN model parameter's optimization after each step. Epsilon-greedy policy [24] is adopted so that the agent can compromise between exploration and exploitation. Adam optimizer is used in the training. The mini-batch size is set as 64. The learning rate is 0.001. The capacity of replay buffer is 10,000. The discount factor is 0.99. On AOD and Grid World tasks, the target Q-value is updated by evaluating Q-value after every 400 and 100 episodes, respectively. The DQN framework is implemented in Pytorch on a NVIDIA GeForce GTX 1080 GPU machine with 8GB of graphic memory.

The evaluation metrics used for AOD task are the same as ones in [20, 21], namely, success rate (SR↑) and average path length (APL↓). The metrics adopted for Grid World task is average score [25] obtained in training process.

4. RESULTS AND DISCUSSIONS

The evaluation results on the AOD task are shown in Table 1. MAP-L and MOP-L are results of MAP and MOP methods using linear policy network. MAP-ours and MOP-ours are results by applying bilinear policy network. From the results, it can be see that MAP-Ours and MOP-Ours outperform MAP-L and MOP-L, respectively, with higher SR and shorter APL, especially on unseen environments. This verifies that the bilinear policy network is more effective in generalizing knowledge learnt from seen environment to unseen environment. Comparing MAP-Ours with MAP-L, performance improvement on seen environment is not significant. But it achieves 9% higher SR and almost the same APL on unseen environment. Comparing MOP1-L and MOP2-L, there is a slight

Table 1. Evaluation results of AOD task on TLESS dataset.

	SR-seen ↑	APL-seen ↓	SR-unseen ↑	APL-unseen ↓
MAP-L	96.2%	3.36	68.3%	5.69
MAP-Ours	**96.6%**	**3.29**	**72.5%**	**5.73**
MOP1-L	82.2%	4.44	62.4%	9.26
MOP1-Ours	**89.9%**	**6.32**	**78.6%**	**8.83**
MOP2-L	81.9%	4.79	55.8%	9.08
MOP2-Ours	**86.1%**	**5.16**	**76.5%**	**8.62**

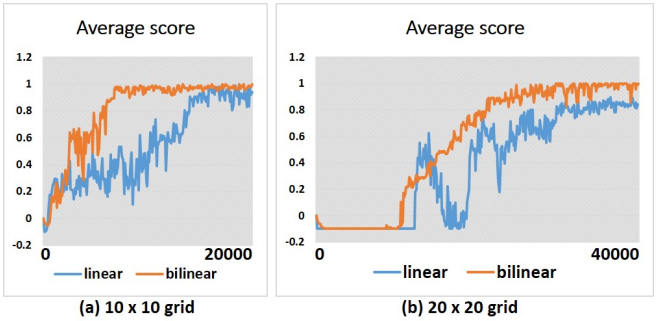


Fig. 5. Average score obtained in the agent training process of Grid World task.

performance drop in SR on seen environment and 7% drop in SR on unseen environment without using extra input as state. This trend also existed between MOP1-Ours and MOP2-Ous. The difference is that, without using extra input, SR drops 2% on unseen environment which is not significant. More importantly, even without using extra input (reduced rely on prior knowledge) in state, the proposed method (MOP2-ours) still outperforms MOP1-L on both seen and unseen environments (especially on unseen environments by 14% higher on SR).

Further comparison between linear and bilinear network is shown as the visualized average scores in terms of the number of training episodes on Grid World task (Fig. 5). In the 10×10 grid (Fig.5 (a)), both methods are converged, but the method with bilinear policy network (our method) converges much earlier and is more stable after converging. In 20×20 grid (Fig.5 (b)), the method with linear policy network cannot converge within specified episodes, while the proposed method can converge within the given episodes. These observations show that bilinear policy network learns the subtle visual representation better than linear policy network, and hence generalizes to unseen environment (different paths to the goal) more effectively.

5. CONCLUSIONS

In this paper, a method using a bilinear policy network as feature extractor in DRL is proposed to effectively learn the representation of subtle visual difference among highly similar states in environments. The bilinear policy network allows pairwise feature interactions in a translationally invariant manner which makes it particularly useful for insignificant difference recognition among highly similar states. Thus, better generalization of the learnt policy from seen environments to unseen environments can be achieved by applying a bilinear policy network in DRL. The effectiveness of the proposed method is demonstrated on AOD and Grid World tasks, which show that the proposed method can achieve better performance especially on unseen environments.

6. REFERENCES

[1] G. Sertan, P. Faruk, and A. Reda, "State similarity based approach for improving performance in rl.," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 01 2007, pp. 817–822.

[2] R. Agarwal, M.C. Machado, P.S. Castro, and M.G. Bellemare, "Contrastive behavioral similarity embeddings for generalization in reinforcement learning," in *International Conference on Learning Representations*, 2021.

[3] P. Hamed, R. Deva, and F. Charless, "Bilinear classifiers for visual recognition," in *Advances in Neural Information Processing Systems*, 2009, vol. 22.

[4] T.Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear cnn models for fine-grained visual recognition," in *2015 IEEE International Conference on Computer Vision*, 2015, pp. 1449–1457.

[5] J. Liu, Z. Yang, T. Zhang, and H. Xiong, "Multi-part compact bilinear cnn for person re-identification," in *IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 2309–2313.

[6] H.Chen, J. Wang, Q. Qi, Y. Li, and H. Sun, "Bilinear cnn models for food recognition," in *International Conference on Digital Image Computing: Techniques and Applications*, 2017.

[7] C. Wang, J. Shi, Q. Zhang, and S.Yin, "Histopathological image classification with bilinear convolutional neural networks," in *International Conference of the IEEE Engineering in Medicine and Biology Society*, 2017.

[8] M. Rekka, K. Nawres, N. Henda, and H.Z. Saoussen, "A bilinear convolutional neural network for lung nodules classification on ct images," *International Journal of Computer Assisted Radiology and Surgery*, 2020.

[9] F. Perronnin, J. Sanchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European Conference on Computer Vision*, 2010, pp. 143–156.

[10] D.G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision*, 1999, pp. 1150–1157.

[11] ZW. Hong, Y. Ge, and P. Agrawal, "Bi-linear value networks for multi-goal reinforcement learning," in *In International Conference on Machine Learning*, 2021, pp. 1038–1044.

[12] B. Scholkopf and A.J. Smola, "Learning with kernels: support vector machines, regularization, optimization and beyond," *MIT press*, 2001.

[13] J. Krause, H. Jin, and L. Fei-Fei, "Fine-grained recognition without part annotations," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[14] X. Zhang, H. Xiong, W. Zhou, W. Lin, and Q. Tian, "Picking deep filter responses for fine-grained image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] L.P. Kaelbling, M.L. Littman, and A.W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, pp. 237–285, 1996.

[16] R. Agarwal, M.C. Machado, P.S. Castro, and M.G. Bellemare, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *In Advances in neural information processing systems*, 1996, pp. 1038–1044.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D.Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[18] S. Zhang and R.S. Sutton, "A deep look at experience replay," in *arXiv preprint arXiv:1712.01275*, 2017.

[19] T. Hodan, P. Haluza, S. Obdrzalek, J.E.S. Matas, M.I.A. Lourakis, and X.Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *IEEE Winter Conference on Aplications of Computer Vision*, 2017, pp. 880–888.

[20] Q. Xu, F. Fang, N. Gauthier, W. Liang, Y. Wu, L .Li, and J.H. Lim, "Towards efficient multiview object detection with adaptive action prediction," in *IEEE International Conference on Robotics and Automation*, 2021.

[21] F. Fang, Q. Xu, N. Gauthier, L.Li, and J.H. Lim, "Enhancing multi-step action prediction for active object detection," in *IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 2189–2193.

[22] X. Han, H. Liu, F. Sun, and X. Zhang, "Active object detection with multistep action prediction using deep q-network," *IEEE Transactions on Industrial Informatics*, pp. 3723–3731, 2019.

[23] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *CoRR*, 2016, vol. abs/1511.05952.

[24] R. Agarwal, "The epsilon greedy algorithm - a performance review," *International Journal of New Technology and Research*, vol. 6, pp. 01–03, 2020.

[25] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," in *In International Conference on Machine Learning*, 2018, pp. 1038–1044.