Combining primitive DQNs for improved reinforcement learning in Minecraft

Matthew Reynard*, Herman Kamper*,
Benjamin Rosman†, Herman A. Engelbrecht*
*Dept. of Electrical and Electronic Engineering, Stellenbosch University, South Africa
Email: {18610943, kamperh, hebrecht}@sun.ac.za
†Dept. of Computer Science and Applied Mathematics, University of the Witwatersrand, South Africa
Email: benjamin.rosman1@wits.ac.za

***Abstract*—We ask whether a reinforcement learning agent learns better by first learning the skills to perform smaller tasks in a complex environment or by learning the skills in the complex environment from the start. This is investigated empirically in a non-trivial game environment. We use the premise of curriculum learning where an agent learns different skills in independent and isolated environments referred to as *dojos* in this paper. The skills learned in the dojos are then used as different actions as the agent decides which skill to perform that best applies to the current game state. We evaluate this with experiments conducted in the Minecraft gaming environment. We find that dojo learning is able to achieve better performance with faster training time in certain environments. The main benefit of this approach is that the reward functions can be finely tuned in the dojos for each action as compared to the traditional methods. However, the skills learned in the individual dojos become the limiting factor in performance as the agent is unable to combine these skills effectively when put in certain complex environments. This can be mitigated if the dojo modules are further trained to result in similar results as the standard method.**

***Index Terms*—Curriculum learning, Reinforcement learning, Minecraft, Project Malmo**

I. Introduction

The aim of having an agent perform well in a challenging environment using reinforcement learning is a long standing goal for researchers. Many ideas, frameworks and algorithms have been proposed. In this work we specifically consider the open-world game of Minecraft.

Minecraft is a popular 3D sandbox game in which players gather resources and build with a variety of blocks in a procedurally generated environment. During the night, mobs (dangerous creatures such as zombies) roam the environment with the players choosing to either avoid them or fight. Surviving in the world of Minecraft requires mastery of various actions, including complex combinations of primitive actions or actions with specific time frames.

In order to survive, our agent will have to master a set of skills and gather knowledge of various aspects of the open-world. Will it be beneficial for the agent to learn solely on its own by being thrown in at the deep end, or should we allow the agent to learn in simpler sub-environments which are known to occur in the complex environment (the complex environment is defined to be all sub-environment tasks in one environment)?

With Minecraft being a popular game worldwide, it has grasped the interest of many researchers. It provides a simple platform where experiments can be conducted in reinforcement learning (RL). Microsoft saw potential in this gaming environment and developed Project Malmo [1] to provide a platform for researchers to explore RL algorithms. There are numerous mini-game environments included in Project Malmo, but the overall goal remains to have an agent perform well in the entire unrestricted world of Minecraft.

The goal of this paper is to compare our new network architecture where an agent learns more complex actions in simpler environments to the current standard of RL. The premise of these independent and isolated training environments, hereafter referred to as *dojos*, stems from humans learning in classrooms. The word *dojo* is a Japanese term directly translating to "place of the way" and is a training facility primarily referring to training of Japanese martial arts.

We begin with an overview of the game of Minecraft in the context of RL and why we use Project Malmo. We then discuss the basics of RL, Markov Decision Processes (MDPs), and Q-learning – a common RL algorithm for gaming environments. We consider a simplified Minecraft environment where an agent must acquire three different skills, namely gathering, avoiding and exploring, in order to succeed. We compare the agent learning these skills separately in different sub-environments and using these learnt skills in the complex environment against learning these skills in the complex environment from the start. In our approach we discuss the network's input and architecture. The results show that our approach is effective and the agent achieves better performance in certain environments, and can achieve at least equal results in all environments tested. Some future work will be discussed to improve on the drawbacks of this method. Lastly, some related work is discussed.

II. Background

A. Minecraft

The open-world game of Minecraft is a popular sandbox game as described in Sec. I. Although there is a main objective in Minecraft, players often choose their own path – from

978-1-7281-4162-6/20/$31.00 © 2020 IEEE

building masterpieces to speed running by finishing the game in the shortest time. This variety of play styles has made it a promising research area especially in the realm of RL.

We use environments based on Microsoft's Project Malmo in the world of Minecraft. This allows us to create a simple environment with multiple sub-tasks to reach the ultimate goal as well as have full control over the design of the environment and the choice of tasks that can be learned independently of one another. This makes Minecraft an ideal world to run our experiments and test our hypothesis that learning simple skills in dojos and applying them in a complex environment is better for an RL agent, as opposed to learning the complex skills from scratch in a complex environment.

B. Reinforcement Learning, MDPs and SMDPs

In the field of probability and statistics, the Markov property expresses that the future is independent of the past, given the present as shown in [2], [3]. The current state of an environment completely characterises the position and condition of all entities and objects within that world. Given an environment with the Markov property, only the current state and an appropriate action is needed to determine the next state.

A Markov Decision Process (MDP) describes an RL environment and is defined by the tuple (S, A, R, P, γ), with S being the set of possible states of that environment, A referring to the set of all possible actions an agent can perform, R is the distribution of the rewards given and is dependent on the state and action performed, P is the transition probability between states, and γ is the discount factor and describes how the agent values future rewards.

In a Semi Markov Decision Process (SMDP), the actions have duration. This is referred to as transition time (τ). In an MDP however, the transition from one state to another is instantaneous. The options framework defines an SMDP [4], and *Dojo learning* is closely related to the options framework. Therefore the theory of SMDPs provide a good basis for dojo learning.

C. Q-Learning

Reinforcement learning is a subsection of machine learning where the agent interacts with an environment, receiving an observation of the environment and a delayed feedback for its actions in the form of scalar rewards. This process can be seen in the RL loop in Fig. 1.

There are numerous methods and algorithms for RL, however we will be focussing on the model-free off-policy algorithm of Q-learning, which is a value-based algorithm using the Bellman optimality equation: $V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s, \pi(s), s') V^{\pi}(s')$ (1). This is then reworked to obtain the Q-learning equation: $Q(s, a) = Q(s, a) + \alpha[R(s, a) + \gamma \max_{a'} Q'(s', a') - Q(s, a)]$ (2) which is an iterative equation updating the Q values of the model [5], [6]. The goal is to maximize the expected cumulative reward.

In Eqs. (1) and (2), s is used for the state of the environment, some equations use o instead to represent the observation of the agent. For this explanation we will assume that the environment is fully observable; the observation the agent receives can represent the entire current state of the environment ($s = o$).

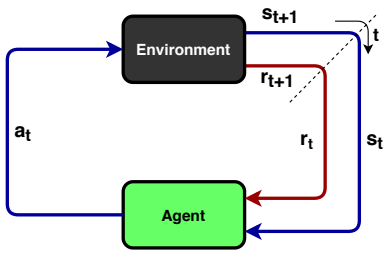
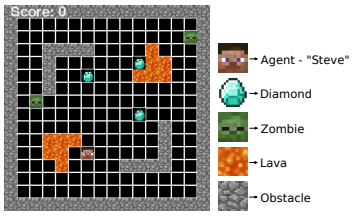


Fig. 1: The Reinforcement Learning (RL) iterative interaction cycle. The Agent performs an action, a_t, given the current state, s_t, of the environment at time t. The Environment then reacts to this action and the Agent receives a scalar reward, r_t, based on the action taken, as well as receiving the next state, s_{t+1}. This cycle is repeated during the learning process.



(a) Minecraft in PyGame - Python version

(b) Project Malmo - Java version

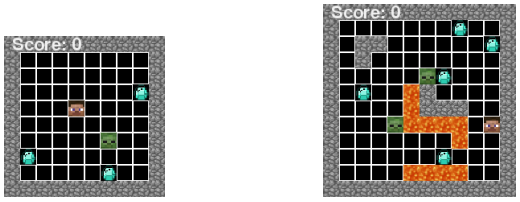
Fig. 2: The same environment represented in both Python's PyGame library and Microsoft's Project Malmo.

We will be using a policy (π) known as epsilon-greedy (ϵ-greedy) for exploration in the environment. This allows our agent to not only make the best action in a certain state, but also random actions to ensure we explore the environment and increase the likelihood that the chosen action is best overall.

III. Environment Setup

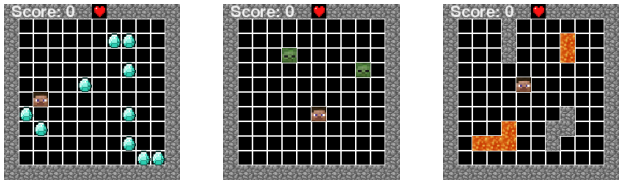
Training an agent in Project Malmo is time intensive due to the platform's complexity. To speed up the research we create the necessary aspects of Minecraft in a Python environment using PyGame [7] for training. The trained network can be transferred to run in Project Malmo in the same environment setup as shown in Fig. 2.

Two types of environments are created, a simple environment represented by Fig. 3a with ten diamonds to collect and one zombie to outrun, and a complex environment represented



(a) Simple Environment (b) Complex Environment

Fig. 3: Two different environments tested for results to show scalability as well as simple versus complex environment behaviour.



(a) Diamond Dojo (b) Zombie Dojo (c) Explore Dojo

Fig. 4: The different dojo environments in which the agent trains in this experiment.

by Fig. 3b with ten diamonds, two zombies and the introduction of obstacles and lava to manoeuvre around.

The agent needs three different skills in order to succeed in the complex environment created, therefore three dojos are used to learn these skills in isolated environments – gathering diamonds (Fig. 4a), avoiding the zombies (Fig. 4b) and exploration (Fig. 4c). For the simple environment only the diamond and zombie skills are used, which is similar to a hierarchical task network (HTN) [8]. One episode consists of a maximum of 100 time steps. For each time step the agent has a choice to do nothing, move up, down, left or right. During the episode, the agent must collect as many diamonds as possible, explore an environment without dying in lava and avoid zombies that move towards it using the A* path finding algorithm [9].

IV. Approach

In the experiments we set out to show the feasibility of our DOJO NETWORK to help an agent perform better in a complex environment. The DOJO NETWORK, shown in Fig. 5, consists of two main parts. The first is the META MODULE. This module decides which dojo skill the agent should reference given the current state of the environment. The second part consists of different DOJO MODULES. These modules are trained independently in a simplified environment and frozen[1], and using transfer learning the trained modules are transferred into the appropriate DOJO MODULES.

The specific deep Q network's architecture used for our experiments is simple but can easily increase in complexity if needed. We show a proof of concept here, and will explore more complex architectures and algorithms in future work.

[1]Frozen: Refers to the model's weights being fixed or unchanged when the rest of the network is being trained.

Each DOJO MODULE has the same network architecture with the same five possible actions as outputs seen in the output layer in Fig. 5: move up, down, left, right or do nothing. However, these actions can be different entirely with one set of output actions from a DOJO MODULE having the primitives[2] to jump over obstacles and another having a set of primitives to craft items. This is combined using a META MODULE with the same main architecture shown in the module block in Fig. 6 and the different dojos as its output. Each time a DOJO MODULE is selected, the appropriate input state (i.e. a state configuration with which that dojo is trained) is then passed over to that trained DOJO MODULE and the best action in that set is chosen. This DOJO MODULE choice occurs at every time step. This approach is similar to the options framework, where one can think of the META MODULE deciding which policy the agent should follow for the current time step.

The network's architecture displayed in Fig. 6 consists of two convolutional layers with two fully connected layers giving the output of five possible actions. Given that using data from raw pixels is computationally expensive, we manually extract the required features for the input state given to the network. The input to the DOJO NETWORK, represented in Fig 7, can be anything from the agents coordinates to the raw RGB values of the screen, as long as the different DOJO MODULES receive the identical input when training. In our case we manually extract the positions of various blocks of interest (BOI) and arrange it in an expandable way. This is done to allow us to easily enlarge the area of interest as well as add complexity by increasing the number of BOIs the model can interpret. A binary-style grid represents the positions of the BOI, with each layer representing a different block type such as cobblestone or lava, or an entity such as a zombie or diamond. The grid is fixed around the position of the agent with odd value dimensions to ensure the midpoint of the grid

[2]Primitives: Basic actions such as move right, jump up, etc.

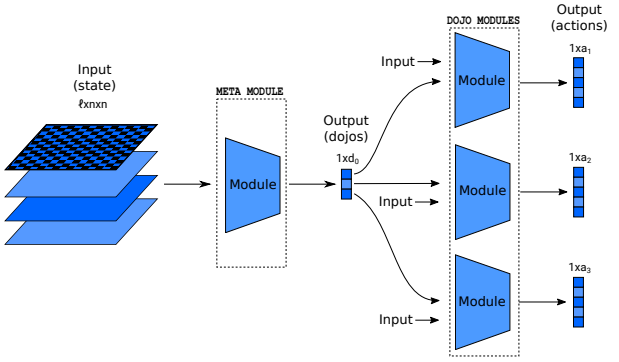


Fig. 5: DOJO NETWORK architecture: This network has three DOJO MODULES, each having its own set of five possible actions. The number of possible outputs for the DOJO NETWORK is the number of unique actions that the DOJO MODULES have. The META MODULE decides which DOJO MODULE will decide on the next action.

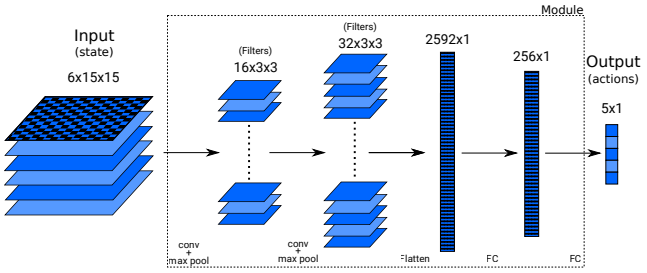


Fig. 6: The network architecture used for DOJO NETWORK and STANDARD Q NETWORK

represents the agent.

V. EXPERIMENTS

We compare the DOJO NETWORK to a traditional Deep Q Network (DQN) [10], which we refer to as the STANDARD Q NETWORK, in the Minecraft gaming environment. These are compared in both a simple and complex environment as shown in the environment setup in Sec. III.

The DOJO MODULES are trained beforehand in the simpler environments shown in Fig. 4 with just the input that affect those particular skills. In the case of gathering diamonds, the dojo is trained without a zombie present. This allows the reward function for that dojo to be refined and suited to that specific scenario. Once all the dojos are trained for 100,000 episodes, the networks are fixed (i.e. no longer trainable) and added to the output of the META MODULE which is trained for a further 100,000 episodes.

After the DOJO NETWORK is trained, with each MODULE having 100,000 training episodes, the DOJO MODULES are unfrozen[3] and the entire DOJO NETWORK is trained for a further 200,000 episodes. The amount of training episodes are an order of magnitude less for the simple environment experiments.

[3]Unfrozen - The weights are no longer fixed and the model is allowed to be trained further.

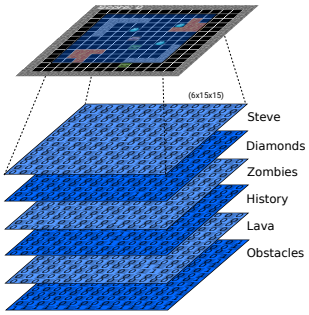


Fig. 7: Network input: Each layer of the 3-dimensional array represents a unique block of interest (BOI) in the grid world environment. A '1' if the BOI occupies this space and a '0' otherwise. The values are placed in relation to the agent's position which is always represented in the center position of the first layer.

TABLE I: Various STANDARD and DOJO network average results in different environments (simple: simple environment; complex: complex environment; new maps: different 10 maps from training; 16: 16 size grid (with 10 different maps))

Network	Moves	Score
Standard (simple)	27.4	7.9
Dojo (simple)	19.9	6.2
Standard (simple - 16)	76.1	5.9
Dojo (simple - 16)	52.6	4.2
Standard (complex)	35.2	8.1
Dojo (complex)	30.6	7.2
Standard (complex - 16)	51.3	5.5
Dojo (complex - 16)	58.0	5.2
Standard (complex - new maps)	33.6	6.3
Dojo (complex - new maps)	31.7	5.8

The model's measure of success is the score received, which is the number of diamonds collected, and moves taken, which can be interpreted as the length of time taken to collect the diamonds or the amount of time surviving the zombies. The models are tested in the ten unique training environments, as well as in slightly different complex environment configurations and sizes which were not seen during training. These are tabulated in the results section, and discussed in Sec. VI.

Also, based on the findings, we conducted two additional experiments. A "cointoss" experiment, which kept the output of the META MODULE random, never pre-trained the DOJO MODULES and simply allowed the entire DOJO network to train for a total of 300,000 episodes. We also conducted an experiment that shows the agent in a zombie free environment. With only the diamond dojo and exploration dojo skills used in the complex environment layout, this is compared to the STANDARD Q NETWORK with the same input. The results of the two additional experiments are discussed in Sec. VI.

VI. RESULTS

The score achieved by the agent is the main value we use to measure the overall success of the different networks. Of all the additional data collected during training and run time, the number of moves taken reveal a key insight to why the STANDARD NETWORK achieved success. The main results of these experiments can be seen in Tab. I.

The STANDARD NETWORK outperformed the DOJO NETWORK in almost every aspect of training. The first set of training time graphs in Fig. 8 show the results of the network being partially "blind" as the input to the DOJO MODULES is the same input used when the modules are independently training. Therefore, once the diamond module is chosen as best skill or "action", the agent has no knowledge of zombies, and the zombies layer in the input is zeroed out[4]. Although looking promising in the simple network (up to 10,000 episodes), the complex network reveals the downfall once the network is unfrozen and the DOJO MODULES are allowed to train at 100,000 episodes. The initial dip in score once unfrozen is expected as the network could have been in a local optimum,

[4]Zeroed out: Every position in that layer is made '0' to reflect a similar training environment.

but without full awareness of the environment, the agent using the DOJO NETWORK fails to reach the score of the STANDARD NETWORK.

A reason for the plateau of the DOJO NETWORK, seen in Fig. 8b, can be due to the fact that it has fixed DOJO MODULES as actions for the META MODULE and cannot make small changes to the DOJO MODULES once exposed to the entire complex environment. In other words, the agent doesn't merge the knowledge of the independent skills, for example it might avoid a zombie or move towards a diamond, but it can't avoid the zombie and move towards a diamond in a single time step.

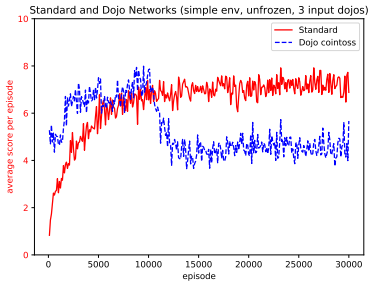
The second set of training time graphs shown in Fig. 9 show the results of the DOJO NETWORK reading the same input as the META MODULE, i.e. the full environment. The results show a lower initial score (compared to the previous results in Fig. 8) due to the confusion of having a new input layer trigger while never being exposed to it in the dojo training. However, once unfrozen, the score climbs to the results of the STANDARD NETWORK, raising the question of whether the DOJO MODULES are simply becoming three identical STANDARD NETWORKS.

These findings led to the "cointoss" experiment which yielded the expected results shown in Fig. 10a. The random choice of the META MODULE action made no difference if all the DOJO MODULES were allowed to train from random initialisations. As well as having a longer training time, as there are three separate networks to train compared to one. This experiment was run to confirm that the worst the DOJO NETWORK could achieve once unfrozen, was at least equal to the STANDARD NETWORK.

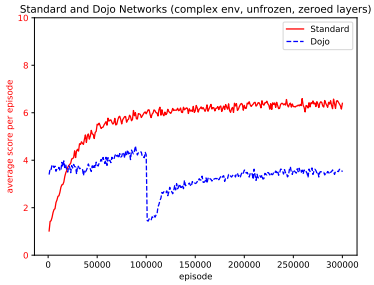
The other additional experiment with the results shown in Fig. 10b is a complex environment setup with no zombies. This proved to be an insightful experiment as the STANDARD NETWORK never reaches the performance of the DOJO NETWORK. It is plausible that the DOJO NETWORK has specific environments in which it outperforms the STANDARD NETWORK based on these results.

VII. Conclusion

We set out to show that an agent that learned a subset of complex actions in a dojo prior to being exposed to the complex environment might outperform an agent which is trained in the complex environment from the start. The evaluated models do not appear to support our hypothesis with the STANDARD NETWORK outperforming our DOJO NETWORK in almost every environment. This is unexpected as a similar hierarchical reinforcement learning approach has been shown to learn Minecraft tasks [11]. Evaluating the results shows the DOJO NETWORK is being limited by the chosen individual modules that are previously learned in isolated dojos and when the agent is exposed to the complex environment it is limited by performing those previously learned actions in a sub-optimal manner. It is possible to match the performance of the STANDARD NETWORK by allowing the DOJO MODULES to be trained further in the complex environment.



(a) STANDARD and DOJO networks, unfrozen at 10k, $\epsilon = 0.1$, simple environment



(b) STANDARD and DOJO networks, unfrozen at 100k, $\epsilon = 0.1$, complex environment

Fig. 8: Dojo input states are limited to observations available when training dojos independently

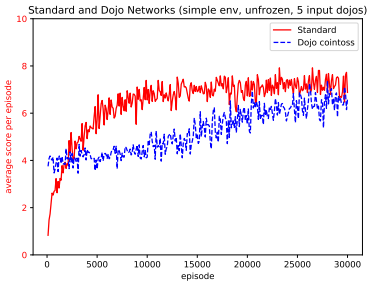
However, it has shown promise that the DOJO NETWORK architecture might be viable in specific environments. Despite the negative results, we believe that by investigating alternative neural network architectures (such as switching models [12] and option-critic architectures [13]), DOJO LEARNING could result in improved performance.

VIII. Future Work

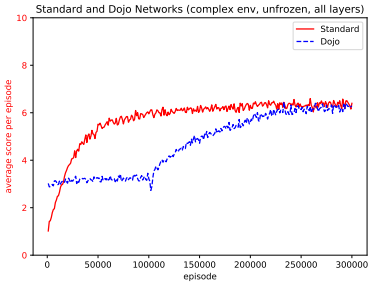
One of the main limitations the agent currently faces is the choice in deciding the skills learnt in the different dojos and then locking the number of skills by predetermining the number of dojos beforehand. We need to improve the agent's method of executing actions and not limit it by our choice in dojos. It will be beneficial to explore a non-fixed action structure to learn about the complex environment in a more efficient manner. We also intend to investigate chosen actions being executed until a certain termination state is reached (or for a specific duration), in a manner similar to the options framework. This could take the form of executing a dojo's learnt policy for multiple time steps instead of a different dojo skill every time step.

We could integrate an additional mosdule or action to the META MODULE, namely the complex action. This action could allow the agent to move and act based on the complex environment when none of the other actions are applicable. This might boost training time and performance, as well as not limit the available actions.

We want to determine in which environments the DOJO NETWORK outperforms the STANDARD NETWORK, and iden-

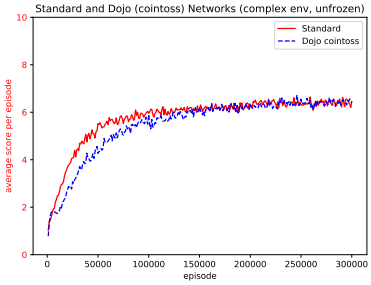


(a) STANDARD and DOJO networks, unfrozen at 10k, ϵ = 0.1, simple environment

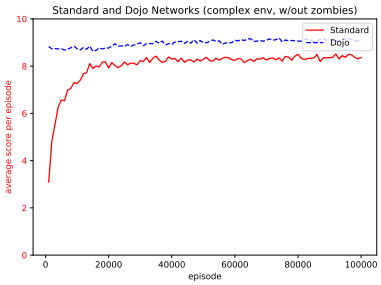


(b) STANDARD and DOJO networks, unfrozen at 100k, ϵ = 0.1, complex environment

Fig. 9: Dojo input states are unlimited and able to observe entire environment



(a) Cointoss experiment, complex environment



(b) No zombies, complex environment

Fig. 10: Additional experiments using the DOJO network as compared to the STANDARD Q network

tify reasons for these results. Finally, we want to include a more complex algorithm and architecture, as mentioned in Sec. IV, and investigate the use of Dueling DQN (DDQN) and prioritized experience replay as shown in [14] and identify the impact of these more advanced algorithms and methods on the DOJO NETWORK.

IX. Related Work

The Options Framework stems from SMDPs in which the transition from one state to another has a time duration and is not instant like in MDPs. Options refer to the combination of primitive actions which may have an extended duration. It consists of a policy (π), a terminating condition (β) and an initiation set (I) [15]. Once an option is chosen, and the state is present in the initiation set, the actions of the agent are decided on by that options policy, and it terminates after a reaching a terminating condition, often being a specified duration.

Curriculum Learning is a special case of transfer learning in which the agent learns smaller, simpler tasks and gradually builds up complexity in tasks in order to increase the performance or learning speed of a more complex task [16]. This method of learning derives directly from the human education system.

References

[1] Microsoft, "Project Malmo," Website, June 2015. [Online]. Available: https://www.microsoft.com/en-us/research/project/project-malmo/

[2] D. Silver, "RL Course by David Silver," 2015. [Online]. Available: "https://www.youtube.com/watch?v=2pWv7GOvuf0\&list=PLzuuYNsE1EZAXYR4FJ75jcJseBmo4KQ9-"

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018.

[4] M. Stolle and D. Precup, "Learning Options in Reinforcement Learning," *School of Computer Science*, 2002.

[5] S. Huang, "Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG)," 2018. [Online]. Available: "https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287"

[6] ADL, "An introduction to Q-Learning: reinforcement learning," 2018. [Online]. Available: "https://medium.freecodecamp.org/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc"

[7] pygame, "Pygame: Free open source python gaming library," 2019. [Online]. Available: "https://www.pygame.org"

[8] K. Erol, J. Hendler, and D. S. Nau, "UMCP: A Sound and Complete Procedure for Hierarchical Task-Network Planning," in *Proc. of the 2nd Intl. Conf. on Artificial Intelligence and Planning Systems*, 1994.

[9] N. Swift, "Easy A* Pathfinding," 2017. [Online]. Available: "https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2"

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 2013.

[11] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A Deep Hierarchical Approach to Lifelong Learning in Minecraft," in *Proc. of the 31st AAAI Conf. on Artificial Intelligence (AAAI'17*. AAAI Press, 2017, pp. 1553–1561.

[12] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic Neural Networks for Hierarchical Reinforcement Learning," in *Proc. of the 5th Intl. Conf. on Learning Representations, ICLR 2017*, 2017.

[13] P.-L. Bacon, J. Harb, and D. Precup, "The Option-critic Architecture," in *Proc. of the 31st AAAI Conf. on Artificial Intelligence (AAAI'17*. AAAI Press, 2017, pp. 1726–1734.

[14] T. Simonini, "Improvements in Deep Q Learning: Dueling Double DQN, Prioritized Experience Replay, and fixed," 2018. [Online]. Available: https://www.freecodecamp.org/news/improvements-in-deep-q-learning-dueling-double-dqn-prioritized-experience-replay-and-fixed-58b130cc5682"

[15] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, 1998.

[16] S. Narvekar, "Curriculum Learning in Reinforcement Learning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 08 2017, pp. 5195–5196.