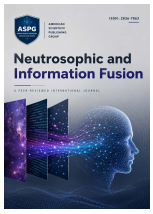
Neutrosophic and Information Fusion

♦ ISSN: 2836-7863 Vol: 03, No. 02, 2024 ♦ PP. 07–11

Article DOI: https://doi.org/10.54216/NIF.030202



FULL LENGTH ARTICLE

On the Effect of the Layers' Number of Deep Neural Network for Improving the Reward of a Reinforcement Learning Robot

Talal Markabi[1,*] **Bahaa Mansoura**[1]

[1] Faculty of Informatics Engineering, Albaath University, Syria

Email: Talal67markabi@gmail.com · mansoura_bahaa77@gmail.com

Received: November 26, 2023 Accepted: March 22, 2024 ⋆ Corresponding author

ABSTRACT

The Q-learning algorithm in reinforcement learning is one of the algorithms that allows the robot to learn the surrounding environment without the need for prior training samples, using the principle of reward and punishment through interaction with the environment. Increasing the number of hidden layers of the deep neural network and adjusting some of its hyperparameters can increase the robot's reward and thus obtain the best path to achieve the goal.

Keywords: Neural network ▪ Deep learning ▪ Robotics ▪ Layers

1. INTRODUCTION

Reinforcement learning is an active research focus in the field of machine learning, and several papers are published annually in this field. At the end of 2013, a small group of researchers from DeepMind released the research paper "Playing Atari with Deep Reinforcement Learning" [1]. A few months later, Google bought DeepMind for a relatively large sum of money. In 2016, Google's DeepMind software AlphaGo defeated South Korean professional Lee Sedol in the well-known game of Go. The terms artificial intelligence, machine learning, and deep learning were then used in the media to describe how DeepMind won, and each is considered part of the reason why AlphaGo won over Lee Sedol.

The Q-learning algorithm in reinforcement learning is one of the best algorithms for unknown environments. The agent receives a reward for a correct step or a penalty for an incorrect step and therefore seeks to accumulate reward at each step to reach the goal in the best possible way. Reinforcement learning [2] differs from other types of machine learning, such as supervised learning, where the agent knows the outputs and inputs as in classification processes, and unsupervised learning, where the agent knows some inputs and arranges them according to their similarity, such as image recognition in social networks.

2. PREVIOUS WORKS

Reinforcement-learning algorithms can be classified into model-free and model-based algorithms. Model-free algorithms include policy-based approaches [3], such as the policy-gradient algorithm. In this approach, the policy function is learned; this function links each state to the appropriate action. Other approaches are value-based: here the agent aims to improve the value function $v(s)$, which gives the largest expected future reward that the agent can obtain in a certain state, as in the Q-learning algorithm.

For model-based reinforcement learning, the goal is to determine the policy that gives the best results and improves the agent's reward. A model is used for transitions between states, such as the Markov decision process (MDP) [4] and the Monte Carlo algorithm. The main difference between model-free and model-based reinforcement learning is that, in the latter, the environment is modeled. This means creating

© 2024 The Author(s). Published by ASPG. This article is licensed under CC BY 4.0.

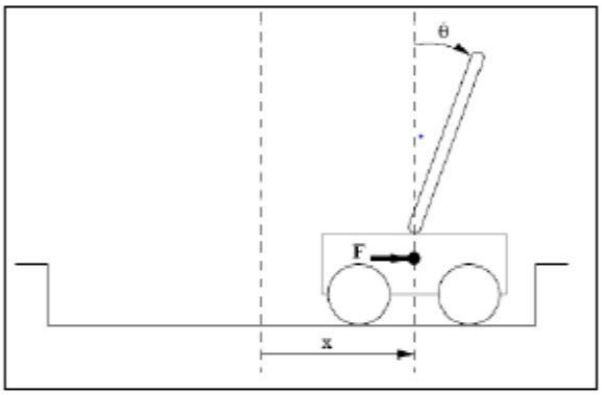


Figure 1. CartPole-v1 environment.

a model of the behavior of the environment surrounding the agent and then determining the policy. This pattern requires modeling a new environment for each studied case, which is difficult in practice.

This study focuses on the use of the model-free Q-learning algorithm [5]. By adding a deep neural network, deep reinforcement learning is obtained and becomes fully based on the neural network. Many types of neural networks support machine learning, including convolutional neural networks (CNNs), used in image and video processing [6], and recurrent neural networks (RNNs), which support audio processing [7]. In this research, the Q-learning algorithm is used in addition to a deep neural network (DNN).

3. RESEARCH TOOLS AND METHODS

A set of libraries was used: TensorFlow, Keras, and NumPy to support deep learning; Matplotlib to draw graphical curves; and Gym, which provides a number of environments in Python for testing results. The CartPole-v1 environment was selected, as shown in Figure 1.

The equations describing its motion are

$$2\ddot{x} + \ddot{\theta}\cos\theta - \dot{\theta}^2 \sin\theta = f_x, \tag{1}$$

$$\ddot{x}\cos\theta + \ddot{\theta} + \sin\theta = 0, \tag{2}$$

where θ is the angle of the pole, x is the cart position, and f_x is the cart traction force.

4. CHARACTERIZATION OF WORK IN THE CARTPOLE-V1 ENVIRONMENT

The elements of reinforcement learning are the agent, the environment in which it moves, the actions it performs, the states, the rewards, and the policy with which it learns to move according to the reward it receives. The elements in the CartPole-v1 environment are as follows:

- **Interaction with the environment:** moving left or right.
- **States:** cart position, cart velocity, pole angle, and pole-tip velocity.
- **Reward:** $+1$ when the pole angle is less than 15° from the balanced position and the cart is less than 2.4 units from the center; -1 otherwise.
- **Policy:** the cart attempts to move right or left to keep the pole stable at an angle of less than 15°.

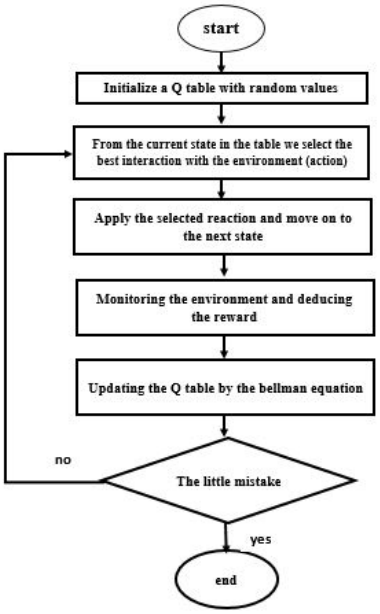


Figure 2. Stages of the Q loop of the cart-pole system.

5. THE Q-LEARNING ALGORITHM

The Q-learning algorithm is one of the most important model-free reinforcement-learning algorithms. It allows the agent to operate very effectively in an unknown environment, accumulating rewards to reach the goal in the best way. Its mathematical formulation is based on the Bellman equation [8]:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\big(R_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\big). \tag{3}$$

The stages of the Q-learning algorithm are shown in Figure 2.

6. DEEP Q-LEARNING ALGORITHM

Deep neural networks have a strong ability to extract complex features. Combining deep learning and reinforcement learning produces deep Q-learning. Deep learning is a branch of machine learning that relies entirely on neural networks; the neural network simulates aspects of the human brain, and deep learning can therefore be regarded as a type of simulation of the human mind.

Because deep learning is supervised learning, it ordinarily needs a training set to return the reward value. Reinforcement learning does not require a predefined training set because the reward is obtained directly from the environment. Combining the two techniques can reduce the need for predefined engineered features and eliminate unnecessary costs. Some disadvantages nevertheless arise from the huge amount of data: the training process has a high cost, requires a long time, and needs powerful processors.

The main difference between Q-learning and deep Q-learning is that the deep algorithm replaces the conventional Q-table with a deep neural network. The network inputs are configured to represent the states of the environment, using the pair consisting of the interaction and the Q-value. The proposed system is shown in Figure 3.

The network inputs are the possible current states of the agent

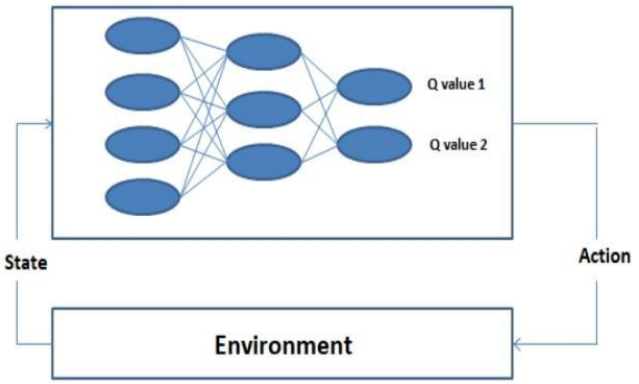


Figure 3. Block diagram of the proposed system after adding the neural network.

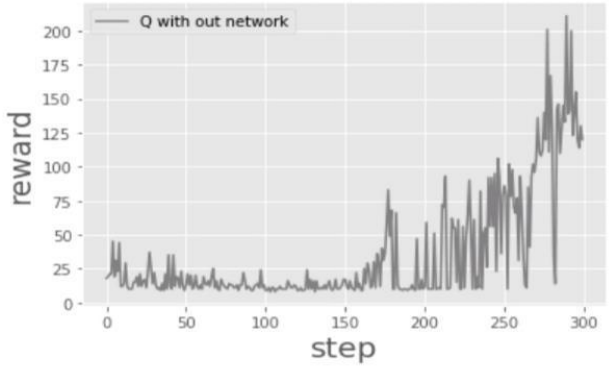


Figure 4. The agent's reward over 300 training cycles using only the Q-learning algorithm.

in the CartPole-v1 environment. There are four states, while the network has two outputs representing the two possible actions: moving the cart left or right. In the second stage of the deep Q-learning algorithm, the appropriate action is selected according to the epsilon-greedy algorithm, discussed later. In the final stage, the weights of the deep neural network are updated according to the Bellman equation.

7. PRACTICAL APPLICATION AND RESULTS

In the first stage, the Q-learning algorithm was applied without adding a neural network. Figure 4 shows the change in the agent's reward after 300 training cycles.

In the second stage, the system proposed in Figure 3 was applied to the CartPole-v1 environment in the Gym library. A neural network with two hidden layers was used, and the numbers of neurons represented decreasing powers of 2. The resulting change in the agent's reward over 300 training cycles is shown in Figure 5.

Figure 5 presents a network configuration of $(4, 32, 8)$: four input nodes and 32 and 8 neurons in the two hidden layers. This arrangement was chosen to improve network performance because processor path lines and memory addresses are represented using powers of 2, in accordance with Andrew Ng's observations [9]. A relative improvement in the agent's reward over the previous stage can be observed.

In the third stage, a deep neural network with three hidden layers was used, again with a decreasing number of neurons represented by powers of 2. The change in the agent's reward after 300 training cycles is shown in Figure 6.

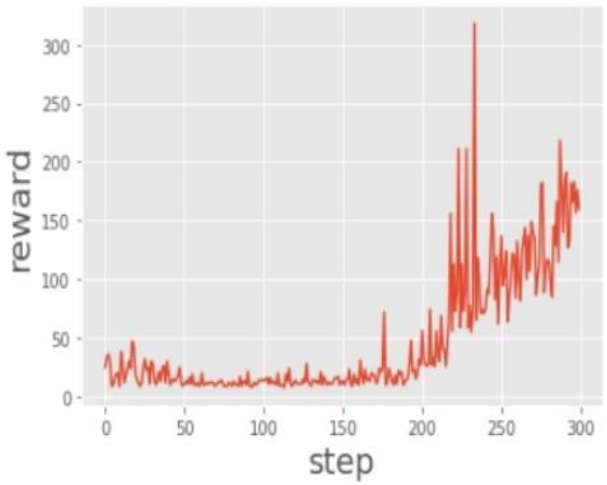


Figure 5. The agent's reward over 300 training cycles with a neural network containing two hidden layers.

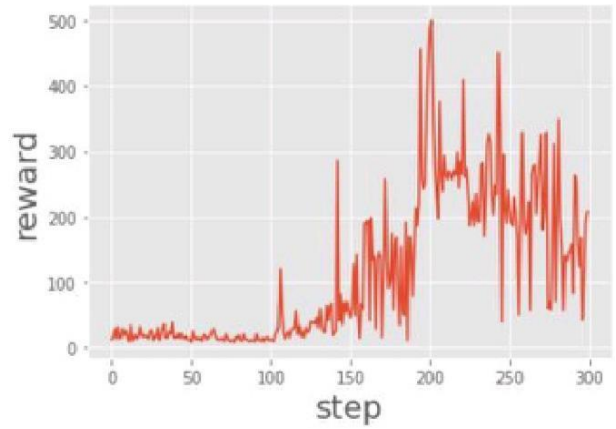


Figure 6. The agent's reward over 300 training cycles with a deep neural network containing three hidden layers.

In Figure 6, three hidden layers with the neuron arrangement $(4, 64, 32, 16)$ were used. An evident improvement of almost 50% over the previous stage is observed. The sharp drops in the result graphs at each stage are caused by the epsilon-greedy technique. During a small, configurable portion of the training cycle, the agent selects random actions to discover new possibilities and then returns to selecting the action with the highest reward.

In the fourth stage, a deep neural network with four hidden layers and the neuron arrangement $(4, 64, 32, 16, 8)$ was used. The change in the agent's reward over 300 training cycles is shown in Figure 7.

Figure 8 compares the reward values of the stages with the application of Q-learning without a neural network. The reward improves clearly after each stage for two reasons: the neurons are arranged using decreasing powers of 2, and the number of hidden layers is increased. These results demonstrate the importance of deep learning in improving the reward.

Increasing the number of hidden layers alone cannot improve the reward to the required extent. Figure 9 compares the third stage of the proposed application with a deep neural network containing three hidden layers and a random neuron arrangement $(4, 14, 25, 77)$. The result illustrates the importance of arranging the neurons in a decreasing order represented by

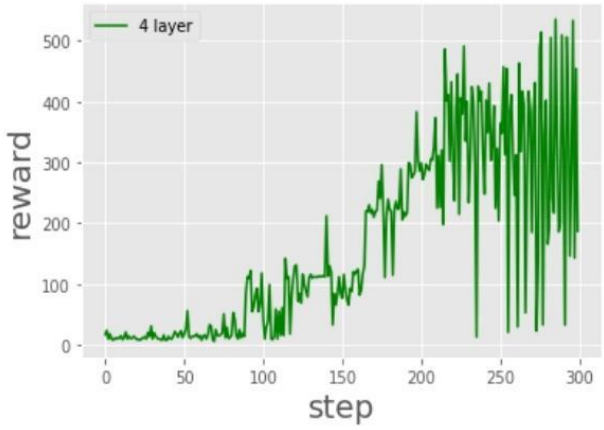


Figure 7. The agent's reward over 300 training cycles with a deep neural network containing four hidden layers.

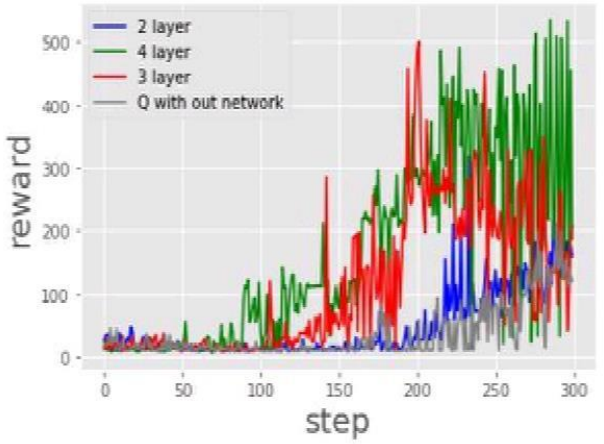


Figure 8. Comparison of the results of the four stages after 300 training cycles.

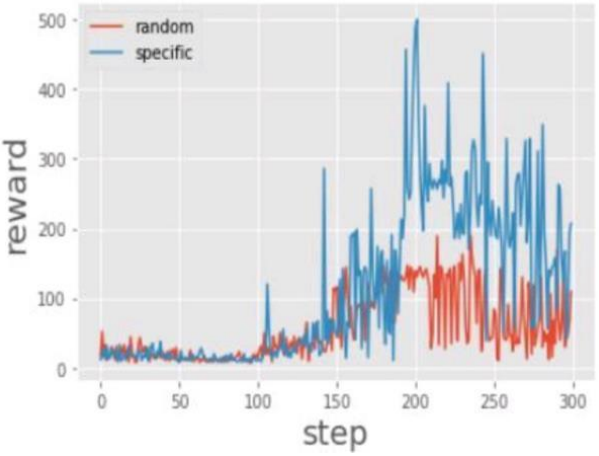


Figure 9. Comparison of the third stage with three hidden layers arranged using random and specified numbers of neurons.

powers of 2. This arrangement makes better use of computer resources because processor path lines and memory addresses are powers of 2.

8. EFFECT OF EPSILON-GREEDY IN THE Q-LEARNING ALGORITHM

The importance of the epsilon factor [10] in Q-learning lies in balancing exploration and exploitation. Always following

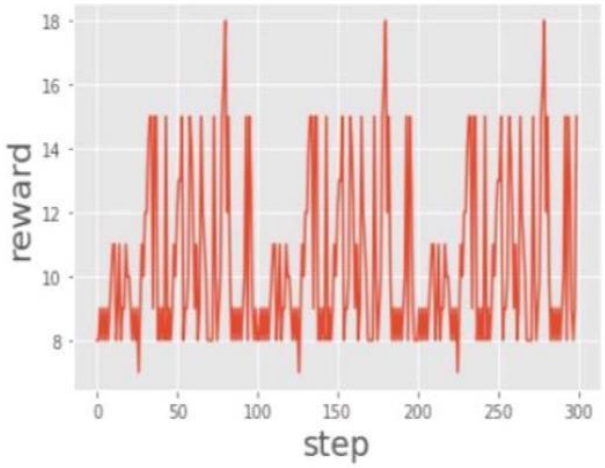


Figure 10. The agent's reward over 300 training cycles with three hidden layers and $\varepsilon = 0$.

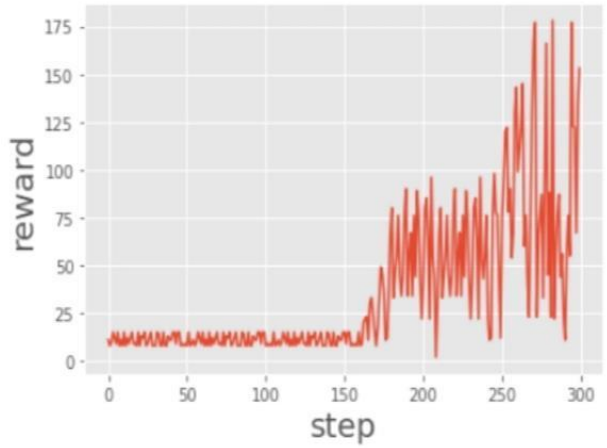


Figure 11. The agent's reward over 300 training cycles with three hidden layers and $\varepsilon = 0.5$.

the path with the highest known reward may be ineffective because the agent does not discover new paths that could lead to better results and higher rewards. The epsilon value can therefore be adjusted so that part of the training time at each step is allocated to exploration and the remainder to exploitation.

When the algorithm operates without exploration, $\varepsilon = 0$, the reward remains nearly constant because no new actions are discovered, as shown in Figure 10.

Figure 10 shows that the reward is fixed at very small values because new actions are not explored. With $\varepsilon = 0.5$, corresponding to 5% of the training time for exploration under the adopted implementation, the response shown in Figure 11 is obtained.

Figure 11 demonstrates the importance of exploration in Q-learning through the clear increase in reward. Figure 6 used $\varepsilon = 1$, the maximum adopted value for exploration, corresponding to 10% of the training time. Figure 12 compares the different epsilon values.

Figure 12 shows that the agent's reward increases as the exploration time increases, demonstrating the importance of the epsilon-greedy technique in the deep Q-learning algorithm.

Finally, a simulation of the CartPole-v1 environment shows the change in the pole angle θ during each training cycle

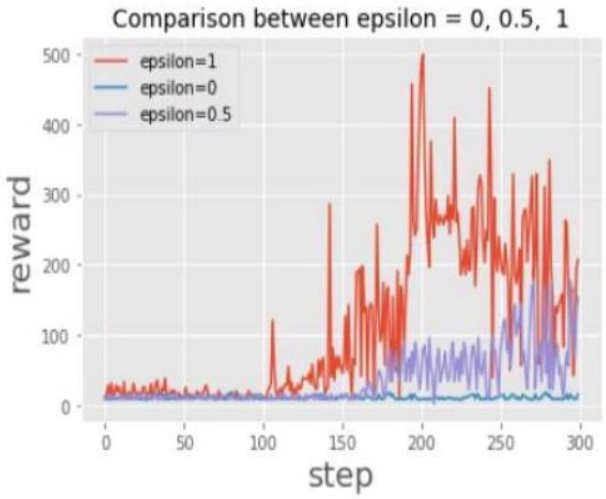


Figure 12. Comparison of epsilon values through the agent's reward over 300 training cycles with three hidden layers.

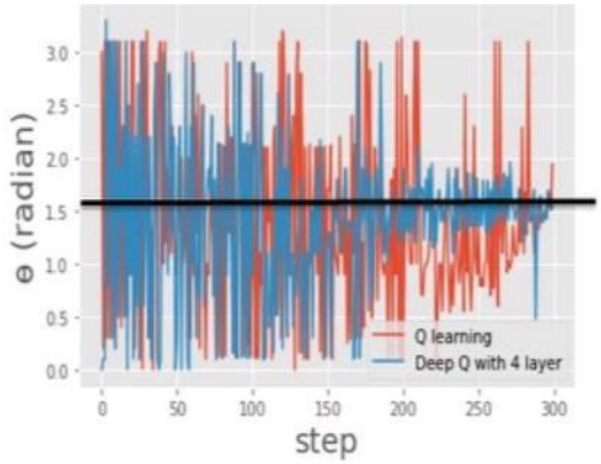


Figure 13. Change in the pole angle during each training cycle.

around the equilibrium position of 90°, or approximately 1.57 radians (Figure 13).

Figure 13 shows that system performance improves after using the deep neural network with four hidden layers. The pole is more stable around the equilibrium position, particularly in the final stages of training.

9. CONCLUSIONS AND RECOMMENDATIONS

Analysis of the results shows an improvement in system performance when the number of hidden layers in the deep network is increased and when the number of neurons follows decreasing powers of 2. This configuration allows better use of computer resources because processor paths and memory addresses are represented by powers of 2.

Other techniques can be used to improve the agent's reward and achieve the desired goal more effectively and quickly:

1. Modular neural networks (MNNs) can be used so that multiple neural networks of different types work together.
2. The proposed system can be applied to a physical agent using a camera, allowing convolutional neural networks to be added for image processing and performance improvement.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," NIPS Deep Learning Workshop, 2013.

[2] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A gentle introduction to reinforcement learning and its application in different fields," *IEEE Access*, vol. 8, pp. 209320–209344, 2020.

[3] S. Aradi, T. Becsi, and P. Gaspar, "Policy gradient based reinforcement learning approach for autonomous highway driving," in *Proc. IEEE Conf. Control Technology and Applications*, Copenhagen, Denmark, pp. 670–675, 2018.

[4] S. Doltsinis, P. Ferreira, and N. Lohse, "An MDP model-based reinforcement learning approach for production station ramp-up optimization: Q-learning analysis," *IEEE Trans. Systems, Man, and Cybernetics: Systems*, vol. 44, no. 9, pp. 1125–1138, 2014.

[5] M. Ronecker and Y. Zhu, "Deep Q-network based decision making for autonomous driving," in *Proc. 3rd Int. Conf. Robotics and Automation Sciences*, pp. 154–160, 2019.

[6] A. Saini, T. Gupta, R. Kumar, A. Gupta, M. Panwar, and A. Mittal, "Image-based Indian monument recognition using convolutional neural networks," in *Proc. Int. Conf. Big Data, IoT and Data Science*, pp. 138–142, 2017.

[7] E. Messner, M. Zohrer, and F. Pernkopf, "Heart sound segmentation—an event detection approach using deep recurrent neural networks," *IEEE Trans. Biomedical Engineering*, vol. 65, no. 9, pp. 1964–1974, 2018.

[8] B. O'Donoghue, I. Osband, R. Munos, and V. Mnih, "The uncertainty Bellman equation and exploration," in *Proc. 35th Int. Conf. Machine Learning*, vol. 80, pp. 3839–3848, 2018.

[9] A. Ng, *Machine Learning*. Stanford University, 2017.

[10] R. Rao and K. Narasimhan, "M-stage epsilon-greedy exploration for reinforcement learning," in *AAAI-21 Workshop on Reinforcement Learning in Games*, 2021.