License to Drive: An RL Agent's First Parking Test

Ajlal Paracha and Alexander Matros

July 31, 2025

Abstract

Artificial Intelligence (AI) and Machine Learning (ML) are viable approaches to the problem of automated vehicle parking [1]. This study offers a comparative analysis of the effectiveness of various Reinforcement Learning (RL) methods in the simulated parking-v0 Gymnasium environment. Specifically, we apply the Advantage Actor Critic (A2C), Proximal Policy Optimization (PPO), and Deep Deterministic Policy Gradient (DDPG) algorithms to train RL agents for the continuous control task of parking using the Stable Baselines3 library, which has implementations for each. Our results found that the off-policy DDPG algorithm successfully learned the task, in contrast to the on-policy methods which failed to converge, highlighting the critical role of sample efficiency in this sparse-reward problem domain. The source code for the implementation of our work can be found at the following GitHub repository: https://github.com/alexmatros/cs486-final-project.

Introduction

Parking a vehicle is a significant source of automotive accidents; one in five accidents occur in a parking lot, and two-thirds of drivers are reported to be distracted while driving in a parking lot [2]. Semi-autonomous, parking-assistance systems have shown noticeable decreases in parking accidents [3]. Thus, the development of fully autonomous parking systems presents the possibility of immense benefit in further reducing parking accidents. The aim of this report is to explore the applicability of various state-of-the-art reinforcement learning algorithms in a simplified parking scenario.

Problem Description

We seek to apply RL algorithms to solve the task of parking a 2D car in a parking lot.

- **Goal:** Park the car into a designated space.
- **Environment:**
  - An empty parking lot.
  - 15 parking spaces in front and behind the car, arranged as 2 rows.
  - One space randomly contains the goal node.
- **Features:** The car's X-and-Y position, X-and-Y velocity, the sin-and-cos steering angle tilts and the goal's X-and-Y position.
- **Actions/Inputs:** The agent controls the car's steering angle and acceleration. This is a continuous action space.
- **Constraints:**
  - The car must always stay within the lot's boundaries.

- The car's final position must be within the goal parking space's lines.

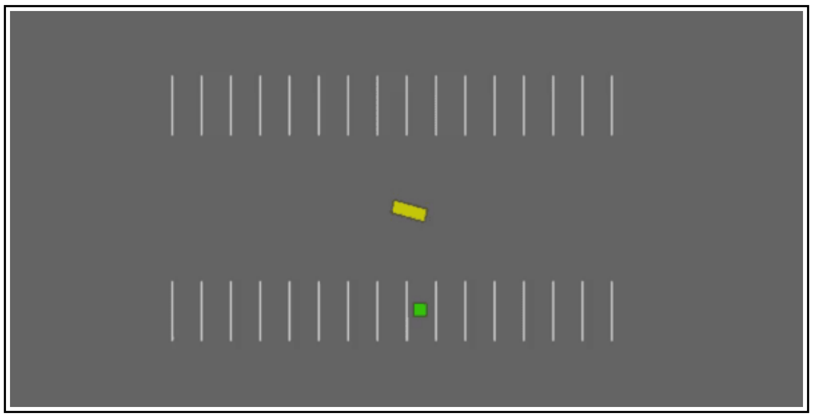

Figure 1: Initial state of the 2D car in the parking-v0 Gymnasium environment. The goal post is randomly placed within one of the 30 parking spaces.

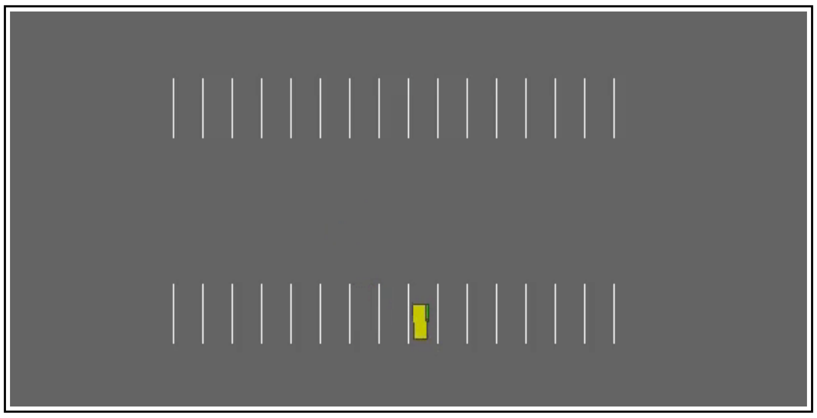

Figure 2: Final goal state of the 2D car in the parking-v0 Gymnasium environment. The car is successfully parked in the goal parking space.

Algorithm Design

This section briefly discusses the theory of Reinforcement Learning (RL) and introduces the Advantage Actor Critic (A2C), Proximal Policy Optimization (PPO), and Deep Deterministic Policy Gradient (DDPG) algorithms.

Reinforcement Learning (RL)

Reinforcement Learning (RL) is a machine learning technique wherein an agent learns to make decisions in an environment by maximizing a cumulative reward to solve a problem [4].

- **Key Components:**
  - **Agent:** The learning model that makes decisions to achieve the goal.
  - **Environment:** The simulation environment, providing the agent states and rewards.
  - **State (s):** A representation of the environment at a given time.
  - **Action (a):** A decision made by the agent in a given state.
  - **Reward (r):** A scalar feedback value from the environment indicating how good an action is in a given state. The agent's goal is to maximize the cumulative reward over time.

  - **Policy (π):** A mapping from states to actions (i.e. the best action for a given state, which is learned).
  - **Value Function $V(s)$:** An estimate of the expected cumulative reward from a given state.

- **Learning Process:**
  - The agent observes the environment's state.
  - As per its policy, it chooses an action to take.
  - Applying the action, the environment transitions to a new state and provides the agent a reward and the state.
  - The agent uses this feedback (reward and new state) to update its policy to improve future rewards.
  - This iterative process is repeated for the specified training iterations.

A given sequence of interactions between the agent and the environment, from some initial state to the end state, is an episode.

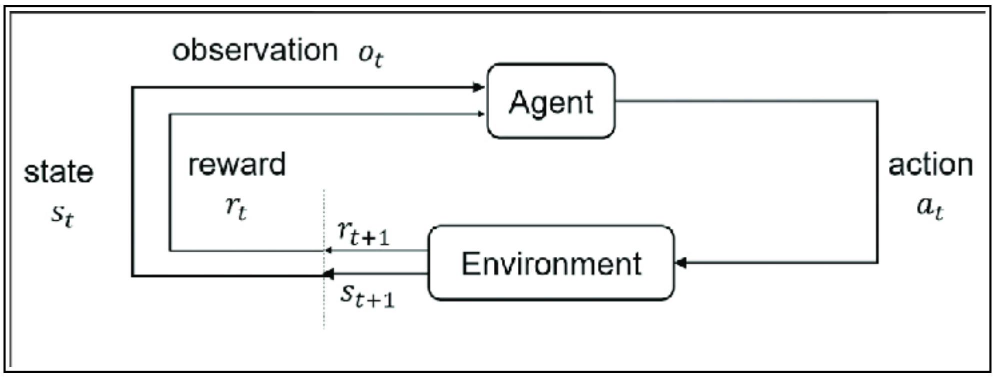

Figure 3: The Reinforcement Learning feedback loop, adapted [5].

Policy Gradient Methods

Policy Gradient methods are a category of RL algorithms that, instead of learning a value function for action selection, directly adjust the policy itself, by estimating the gradients of the expected reward with respect to the policy parameters [6].

- **Objective:** Maximize the objective function, $J(\theta)$, which is the expected cumulative reward of the policy π_θ. The optimal policy is learned using gradient ascent (the maximizing dual of gradient descent):
$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k)$$
where α is the learning rate.

- **Key Characteristics:**
  - **Continuous Actions:** They work well in environments with continuous action spaces, like the steering and acceleration required for parking.
  - **Stochastic Policies:** They learn stochastic probabilities which help with exploration by motivating exploration and trying different actions.
  - **High Variance:** Because of the possibility for high variance in gradient estimates, they can have an unstable learning process.

A2C and PPO are advanced policy gradient methods that introduce functionality to counter the high variance problem.

Advantage Actor Critic (A2C)

A2C is an on-policy, actor-critic algorithm that employs ”advantage” to improve the stability and performance of policy gradient methods. It is a synchronous version of Asynchronous Advantage Actor-Critic (A3C) [7].

- **On-Policy:** Learns from data collected by the current policy.
- **Actor-Critic Architecture:**
  - **Actor:** Learns a stochastic policy (π), which outpus a probability distribution over actions.
  - **Critic:** Learns a state-value function ($V(s)$).
- **Advantage Function ($A(s,a)$):** A2C uses the advantage function, which estimates the relative performance of an action. It determines how much better an action is compared to the average action expected under the current policy. This reduces variance, thus enabling more stable learning. A common advantage function is the Temporal Difference (TD) error:

$$A(s,a) = r + \gamma V(s') - V(s)$$

- **Synchronous Training:** The actor and critic networks in A2C are trained synchronously with a shared optimizer.

Proximal Policy Optimization (PPO)

PPO is an on-policy, actor-critic algorithm with a focus on performance and training stability. By applying the policy gradient method to optimize a clipped objective function, the policy updates are smaller, and thus more stable [8].

- **On-Policy:** Learns from data collected by the current policy.
- **Actor-Critic Architecture:**
  - **Actor:** A neural network that learns the policy (π), mapping states to actions.
  - **Critic:** A neural network that learns the value function ($V(s)$), estimating the expected return from a given state.
- **Clipped Objective:** PPO introduces a clipping mechanism to its objective function, to enable smaller, more stable policy updates during optimization.

Deep Deterministic Policy Gradient (DDPG)

DDPG is an off-policy, actor-critic algorithm that extends Deep Q-Networks (DQN) to continuous control problems [9].

- **Off-Policy:** Can learn from experiences generated by a different policy than the one currently being optimized (enabling experience replay).
- **Actor-Critic Architecture:**
  - **Actor:** A neural network that learns a deterministic policy (π), directly outputting the best action for a given state.
  - **Critic:** A neural network that estimates the Q-value of a state-action pair ($Q(s,a)$), representing the expected return from taking a specific action in a given state.

- **Target Networks:** DDPG uses separate target networks for both the actor and critic. These target networks are updated slowly (soft updates) from the main networks, which helps stabilize training by providing more stable target Q-values for the critic.
- **Experience Replay Buffer:** Stores past transitions (state, action, reward, next state) and samples them randomly to train the networks. This breaks correlations between consecutive samples, improving learning stability.
- **Exploration Noise:** Since the policy is deterministic, exploration during training is achieved by adding noise (e.g. Gaussian noise) to the actions chosen by the actor.

Methodology

We evaluated the performance of Advantage Actor Critic (A2C), Proximal Policy Optimization (PPO), and Deep Deterministic Policy Gradient (DDPG) algorithms within the parking-v0 Gymnasium environment.

All models were implemented using the Stable Baselines3 library and employed a MultiInputPolicy. Each algorithm was successively trained for each of 10,000, 50,000, and 250,000 timesteps to observe learning progression across varied training times.

The following common hyperparameters were used for all three algorithms:

- Discount Factor (γ) = 0.98 (to enable far-sightedness, and optimize for the end goal)
- Batch Size = 256 (except A2C, where it was not applicable)
- Learning Rate = 1×10^{-3}

Algorithm-specific hyperparameters were configured as follows:

- **A2C:** The number of steps to run for each environment per update was set to 5.
- **DDPG:** A NormalActionNoise with a mean of zero and a standard deviation of 0.1 was applied for exploration during training.

For all other hyperparameters not explicitly mentioned, the default values provided by the Stable Baselines3 library implementations were utilized.

Experimental Results

The performance of the three reinforcement learning algorithms varied significantly. The Deep Deterministic Policy Gradient (DDPG) model was the only one to successfully learn the parking task, achieving a high success rate. Both the Advantage Actor Critic (A2C) and Proximal Policy Optimization (PPO) models failed to converge to an optimal policy, consistently crashing the vehicle. The detailed metrics are presented in Table 1 and visualized in Figures 4 and 5.

Model	Timesteps	Success Rate (%)	Crash Rate (%)	Timeout Rate (%)	Avg Return	Std Dev Return	Avg Success Return	Avg Ep Length
A2C	10000	1.0	99.0	0.0	-12.20	1.89	-3.52	15.70
A2C	50000	0.0	100.0	0.0	-13.35	2.04	0.00	17.98
A2C	250000	0.0	100.0	0.0	-12.91	2.07	0.00	17.12
PPO	10000	0.0	100.0	0.0	-22.49	11.58	0.00	37.26
PPO	50000	1.0	99.0	0.0	-11.97	1.69	-3.75	15.37
PPO	250000	1.0	99.0	0.0	-11.59	1.68	-2.27	14.62
DDPG	10000	1.0	98.0	1.0	-15.22	25.22	-4.05	23.23
DDPG	50000	55.0	43.0	2.0	-12.42	21.74	-6.85	30.54
DDPG	250000	83.0	16.0	1.0	-8.17	6.80	-6.47	23.50

Table 1: Performance Comparison of RL Algorithms in Parking-v0 Environment with 100 runs.

- **Model:** The reinforcement learning algorithm used.
- **Timesteps:** The total number of simulation steps the model was trained for.
- **Success Rate (%):** The percentage of episodes where the car successfully parked.
- **Crash Rate (%):** The percentage of episodes where the car crashed.
- **Timeout Rate (%):** The percentage of episodes that ended due to reaching the maximum episode length without success or crash.
- **Avg Return:** The average cumulative reward obtained per episode.
- **Std Dev Return:** The standard deviation of the cumulative reward, indicating variability.
- **Avg Success Return:** The average cumulative reward specifically for successful episodes.
- **Avg Ep Length:** The average number of steps taken per episode.

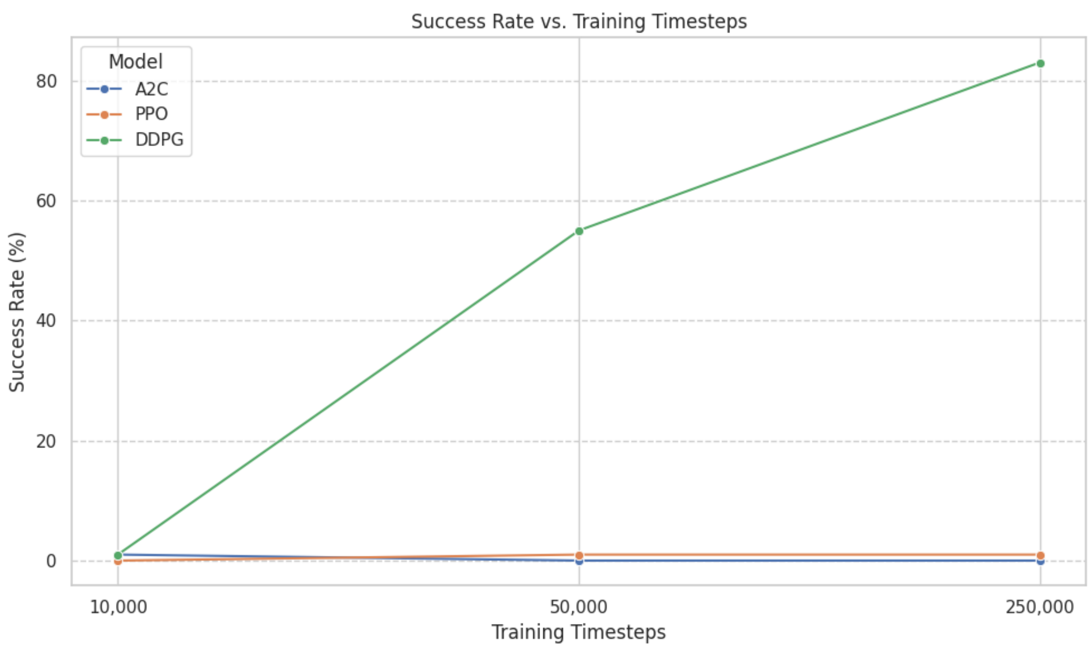

Figure 4: Success Rate vs. Training Timesteps.

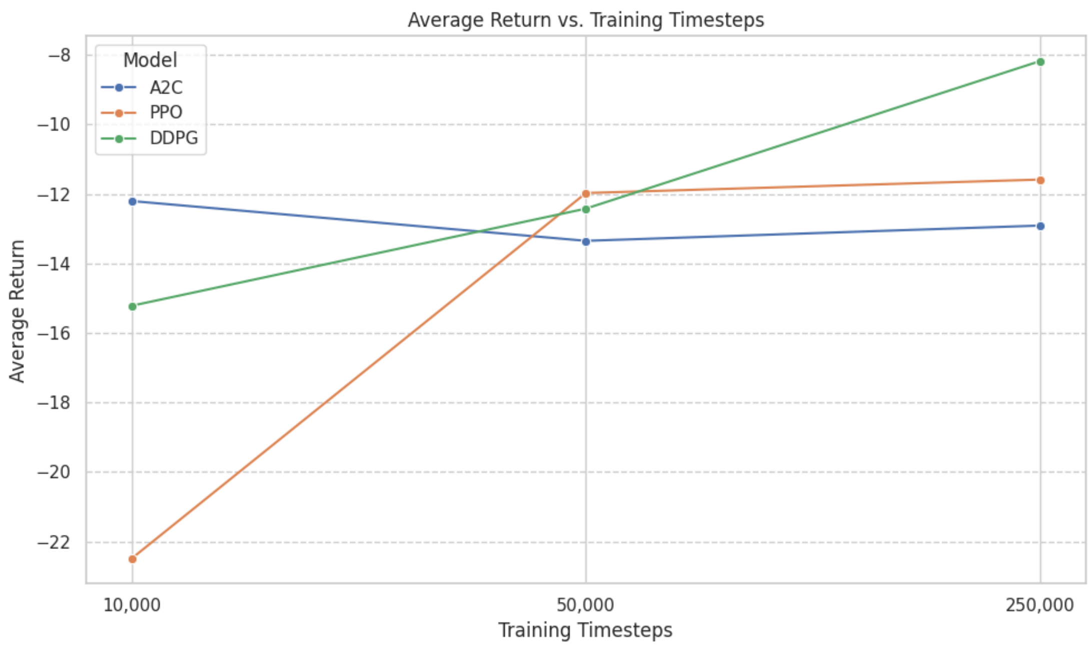

Figure 5: Average Return vs. Training Timesteps.

Analysis

A2C and PPO: Failure due to On-Policy Inefficiency

Both A2C and PPO failed to develop a successful parking policy.

- **Performance:** The A2C model's success rate was 1% at 10,000 timesteps before falling to 0% at both the 50,000 and 250,000 timestep marks. The PPO model started with 0% success at 10,000 timesteps and then plateaued at a mere 1% success rate for both 50,000 and 250,000 timesteps. As such, both models had crash rates between 99% and 100% during the training. While their average returns improved slightly over time, this is little indication that much progress towards learning an optimal parking policy was made.

- **Analysis:** The on-policy nature of A2C and PPO, which learn directly from the current policy's experience without a replay buffer, explains their poor performance. This is known as catastrophic forgetting for on-policy methods [10], as they learn only from the most recent experience, and the success states previously encountered are not saved, thus in a given episode, the success states are very limited, which is another way that the positive reward signal is weakened. Due to the sparse rewards in this exploration task, there is not enough of a positive reward signal in a given training episode to guide the policy meaningfully towards a successful outcome, especially given the 30 distinct parking spaces that the goal may be in. Furthermore, the A2C and PPO methods use stochastic policies which, in a large environment with sparse rewards like parking-v0, means their random exploration is unlikely to encounter the goal state, because of the specific and long-sequence of control actions required to get to a goal state. The agents thus never learn how to successfully park.

  Interestingly, the average returns for PPO have a significant improvement, from -22.49 to -11.97 to -11.59. This indiactes that the PPO agent learned a suboptimal policy of not-losing; that is, as corroborated by empirical viewings, the agent learns to avoid crashing (i.e. exiting the parking lot boundary), rather than how to learn parking.

DDPG: Successful Learning through Off-Policy Exploration

The DDPG agent has significant learning progress.

- **Performance:** The success rate grew steadily from 1% at 10,000 timesteps to 55% at 50,000 timesteps to 83% at 250,000 timesteps. As such, the crash rate fell from 98% to 43% to just 16%.

- **Analysis:** We suspect DDPG's success to be due to its off-policy nature and the use of an experience replay buffer. By storing and randomly sampling past transitions, the agent can learn from a diverse set of experiences, breaking the correlation between consecutive steps. This allows the agent to learn from rare successful episodes multiple times, reinforcing the correct policy and overcoming the sparse reward signal. This property is known as sample-efficiency [9], which is vital in the parking-v0 environment where successful outcomes are initially rare. This stability is further enhanced by DDPG's use of target networks, which provide a stable Q-value for the critic to learn against. Further, the deterministic policy and the action noise to guide exploration make DDPG well-suited for the continuous control problem of parking.

  Further indication of the learning progress is the decrease in the standard deviation of the average return. It started very high at 25.22 to 21.74 down to 6.80 as the agent trained. This change shows the policy shifting from an unstable, high-variance exploration phase to a more stable policy that can successfully park the car across a variety of situations.

Limitations and Future Experimentation

While the trained DDPG model shows promising results, our experimentation has several key limitations:

- **Hyperparameter Choice:** The choice of hyperparameters explicitly mentioned were chosen as per general recommendations, while the rest were set to the default values, and were thus not fine-tuned. Ideally, the models should undergo a process of grid search to find the best hyperparameters. This was skipped due to time limitations, but future experiments should explore this further for unexploited benefits.

- **Model Complexity:** The model training ran only for 250,000 timesteps which took about 30 minutes on our available compute infrastructure (an average laptop, by 2025 standards). Further, the input features are quite basic and bare bones. In future experiments, the training should ideally feature a larger model with more training and more realistic input features, like sensor inputs that replicate a real-world vehicle. This would create a more powerful, robust, and realistic model.

- **Idealistic Parking Lot:** The parking lot is empty within an episode, and the agent will often go over many parking spaces to get to the goal space. In a real parking scenario, with the multitude of other parked cars and moving obstacles, such behavior is undesireable. Thus, to more accurately train a parking agent, future experimentation should feature a more complex and robust environment for training.

- **Exploration of Other Algorithms:** Within RL, there are many more algorithms, like Soft-Actor-Critic (SAC) and Twin Delayed Deep Deterministic Policy Gradient (TD3). Since the successful model was DDPG, comparing it against its direct successor, TD3, would be a logical next step. Furthermore, future experiments could investigate entirely different families of algorithms, such as model-based RL, which can offer greater sample efficiency in certain tasks.

Conclusion

This project presented a comparative analysis of 3 state-of-the-art RL algorithms: A2C, PPO, and DDPG, for the continuous control task of autonomous parking. The results had stark performance differences, highlighting the key difference between off- and on-policy algorithms, with the off-policy algorithm DDPG successfully learning the task achieving an 83% success rate after training for 250,000 timesteps, while A2C and PPO failed to converge on a useful policy. These findings demonstrate the importance of an experience replay buffer, which provides the superior sample efficiency needed in a sparse-reward environment like parking-v0.

Overall, this experience helped solidify the course content on Reinforcement Learning and provided an invaluable opportunity to apply the algorithms and see first-hand their successes and limitations. We are excited for continued work in the field and future projects.

References

[1] A. Raza and M. Z. Yusoff, "Advanced machine learning-driven automated multi-level parking system," in *2024 IEEE Symposium on Wireless Technology Applications (ISWTA)*, pp. 1–6, 2024.

[2] Transline Inc., "Parking lot accident statistics." TranslineInc.com, 2024. Accessed: Jul. 30, 2025.

[3] J. B. Cicchino, "Real-world effects of rear automatic braking and other backing assistance systems," *Journal of Safety Research*, vol. 68, pp. 41–47, 2019.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA, USA: MIT Press, 2nd ed., 2018.

[5] Scientific Figure on ResearchGate, "Deep Reinforcement Learning for Constrained Field Development Optimization in Subsurface Two-phase Flow - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-reinforcement-learning-feedback-loop-adapted-from-30$_fig1_3$53665868,$''$ 2025. $[accessed 31 Jul 2025]$.

[6] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems 12*, pp. 1057–1063, 2000.

[7] V. Mnih, A. P. Badia, K. Babuschkin, M. Fortunato, D. Kurel, Z. Małachowski, A. Sowmya, G. Farquhar, J. Springenberg, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, pp. 1928–1937, PMLR, 2016.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[10] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *The psychology of learning and motivation*, vol. 24, pp. 109–165, Academic Press, 1989.