RL-Trade: Stock Trading using Deep Reinforcement Learning

Saiprashant Vaka
University of Michigan
saivaka@umich.edu

Evan Zimmerman
University of Michigan
evanzimm@umich.edu

I. Introduction

This project explores applying deep reinforcement learning (DRL) to create a stock trading agent that maximizes portfolio returns while adapting to stochastic and complex financial markets. Our work combines recent research in actor-critic algorithms and ensemble strategies to address challenges associated with decision-making in dynamic environments with continuous action spaces. The motivation for this work stems from limitations in traditional trading methods, which rely on static models and are influenced by human biases. Using DRL, this project develops DRL agents capable of making data-driven trading decisions, mitigating emotional decision making, and improving risk management.

Our DRL stock trading agents observe daily stock price data, technical indicators (e.g. MACD, RSI, CCI, ADX), and volatility measures (e.g., VIX). Considering an observation, our agents output an action to buy, sell, or hold shares of each stock it trades on. The primary challenges of developing such DRL stock trading agents include designing the environment, specifically the continuous action space, managing the stochasticity of market data in the observation space, and developing an ensemble agent that can effectively aggregate the actions of multiple DRL trading agents.

Contributions of this project include:

- Data preprocessing to collect stock prices with the Yahoo finance API and calculate relevant technical indicators for the stocks in consideration
- Designing a custom stock trading environment to simulate day-to-day trading.
- Training five different actor-critic DRL algorithms to maximize portfolio return
- Implementing a weighted average ensemble agent and a machine learning meta ensemble agent that aggregates actions from multiple DRL agents.

Each team member has contributed to all components of the project, including initial research, implementation of environment and agent, and performance evaluation.

II. Related Work

A. Reinforcement Learning for Stock Trading

Research by Yang et al. explored an ensemble trading strategy combining three deep reinforcement learning algorithms: Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and Deep Deterministic Policy Gradient (DDPG) [1]. Their approach focused on maximizing portfolio returns, and addressed challenges like transaction costs, liquidity constraints, and market volatility. Their ensemble method entails selecting the agent with the highest Sharpe ratio over a recent validation period. Yang et al. 's research influenced our project's methodology by highlighting the need for an ensemble approach, particularly in that different agents may be better suited for different market conditions.

Bertoluzzo and Corazza focus on creating financial trading systems (FTSs) based on reinforcement learning algorithms like Temporal Difference Q-Learning and Kernel-based Reinforcement Learning [2]. They explore state-action Q-value functions, policy improvement processes, and the practical integration of buy, sell, and hold signals. By applying their methods to both artificial and real financial datasets, they display the importance of using adaptive learning rates and state transformations for better performance in non-stationary environments. While their work primarily addresses discrete state spaces and simpler decision frameworks, their approach highlights how stock trading can fundamentally be modeled as a reinforcement learning problem.

B. Ensemble Strategies in Reinforcement Learning

The survey by Song et al. emphasizes the potential of applying ensemble strategies to reinforcement learning to combine the outputs of multiple agents or models based on their performance [3]. Some ensemble strategies presented in the survey address that ensemble contributions should depend on how individual models demonstrate success across different states or tasks. These strategies that capture this nuance include weighted aggregation and stacking, which involves using additional machine learning to further predict the results of base learners. Such strategies could be particularly advantageous in dynamic environments, such as financial markets.

C. New Contributions of this Project

Building on the foundational research, our work introduces several innovations:

- RL Algorithms: Beyond the three algorithms used by Yang et al. [1], we explore the soft actor-critic algorithm (SAC) and Twin Delayed DDPG algorithm (TD3), for building an effective trading agent.
- Ensemble strategies: We also explore a weighted average ensemble agent and a machine learning meta ensemble agent, inspired by the ensemble strategies highlighted by Song et al [3].

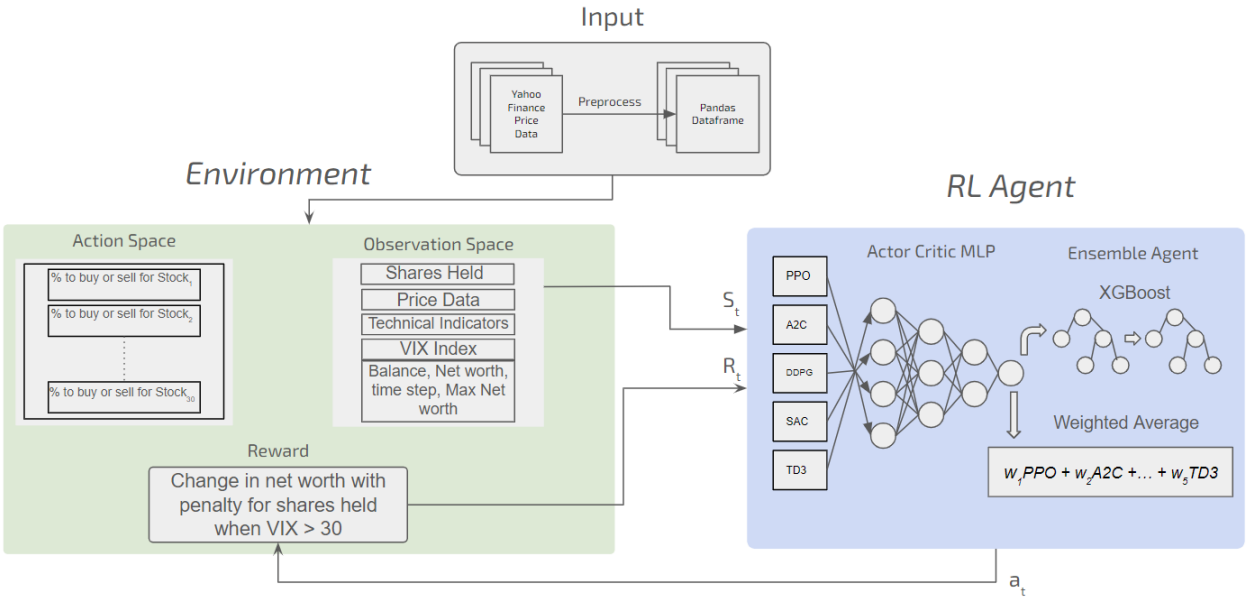

Fig. 1. An overview of the reinforcement learning system architecture designed to train and evaluate trading agents to maximize portfolio return by making daily trades.

- Risk-Adjusted Reward Function: We devised a reward function that balances returns and risk, using the VIX index to add a risk penalty in volatile market conditions.

These contributions aim to intersect the ideas of existing work [4, 5, 1, 6], providing a more adaptable approach to developing DRL stock trading agents.

III. Methodology

Our project uses deep reinforcement learning (DRL) to optimize stock trading strategies in dynamic financial markets. DRL is suited for this task, as it allows an agent to navigate the complex action space of buy, sell, or hold decisions across multiple stocks, with the objective of maximizing portfolio returns. By iteratively interacting with a simulated trading environment, an agent can learn to balance exploration of new strategies with exploitation of profitable ones, guided by rewards based on portfolio performance and risk-adjusted penalties during high market volatility.

The architecture of our reinforcement learning system, shown in Fig. 1., is composed of:

- **Dataset**: Dow jones 30 price data with technical indicators.
- **Environment**: Simulated stock market that the agent interacts with. Manages state transitions and reward calculations.
- **Agent**: Actor-Critic RL algorithms for decision-making.
- **Ensemble Module**: Aggregates individual agent predictions.

A. Datasets

For this project, we utilized stock market data from the Dow Jones 30. The dataset includes daily stock price data spanning from January 2009 to May 2022, approximately 3,500 trading days, and contains features such as *Open*, *High*, *Low*, *Close*, *Adjusted Close*, and *Volume* for each stock. We consider daily data to simplify the trading process for agents such that they can make trades once a day upon observing daily data for a given day. These price features were used to calculate technical indicators, which were added on to the dataset as complementary features. The Technical indicators (*MACD*, *Signal*, *RSI*, *CCI*, and *ADX*) describe the direction and strength of trends in price movements.

Due to its long time horizon, the dataset includes a balanced representation of different market scenarios, making it well-suited for training and testing reinforcement learning agents over varied market conditions. The data set was divided into the training set (01/30/2009 - 12/30/2016), the validation set (01/30/2017 - 12/31/2017), and the test set (01/30/2018 - 05/08/2022). The training set was chosen to include numerous samples of favorable (2012-2015) and adverse (2009) market conditions. The validation set was chosen to reflect relatively calm market conditions for tuning the weighted average ensemble agent. Similar to the training set, the test set was chosen to reflect various market conditions (i.e. covid crash) to effectively evaluate how the agents performed through market duress. To provide agents with a sense of market volatility at a given time, daily VIX index data over the same time ranges was also included in the datasets.

B. Preproccesing

The dataset was sourced from Yahoo Finance using the yfinance library. For each stock, after the price data was downloaded, the data was then formatted into a pandas dataframe, and saved as a csv file to avoid redownloading data for training. Each price data frame was then segmented into a training, validation, and test set according to the dates described in Section 3.A. Once all price data frames

were split, we calculated technical indicators for the training, validation and test set of each stock and added them to their respective dataframes. Each row of these dataframes now corresponded to one day of stock price information & technical indicators, providing a consistent feature structure for the reinforcement learning agents to process during training and testing. Dataframes for each stock over a respective period were packaged together in a python dictionary and passed into the environment upon initialization to grant the environment simple access to stock data.

C. Environment

The environment extended the Gymnasium framework [7] because it provided an easily customized foundation for the action space, observation space, and reward function:

1) Input and Initialization: Processed daily features for each stock and daily VIX data was passed into the environment, and stored by the environment throughout it's lifetime. The balance of the agent was initialized (set to \$10,000), as well as the net worth, maximum net worth, and shares held for each ticker (set to 0), and the transaction cost percentage (set to 0.5%).

2) Observation Space: The observation space is a vector that contains market data, technical indicators, and information related to the agent's portfolio. Let $\mathbf{o}$ represent the observation vector, which consists of the following components:

$$\mathbf{o} = [\mathbf{p}, b, \mathbf{s}, w, \hat{w}, t, v],$$

where:

- $\mathbf{p} = [p_1, p_2, \ldots, p_{30}] \in \mathbb{R}^{30}$ is the price data and technical indicators for all 30 stocks, with p_i representing the price data and technical indicators of the i-th stock.
- $b \in \mathbb{R}$ is the current account balance of the agent.
- $\mathbf{s} = [s_1, s_2, \ldots, s_{30}] \in \mathbb{N}^{30}$ is a vector containing the number of shares held for each of the 30 stocks, with s_i representing the number of shares of the i-th stock.
- $w \in \mathbb{R}$ is the agent's current net worth, which includes both the account balance and the value of the stocks held.
- $\hat{w} \in \mathbb{R}$ is the maximum net worth observed by the agent.
- $t \in \mathbb{N}$ is the current time step in the simulation.
- $v \in \mathbb{R}$ is the VIX index, which measures the volatility of the market.

3) Action Space: The action space is 30-dimensional, where each dimension represents an action for a specific stock. The value in each dimension is a continuous element in the range $[-1, 1]$, where:

- A value less than 0 indicates % of current shares to sell,
- A value greater than 0 indicates % of balance to spend on buying shares,
- A value equal to 0 indicates hold.

Thus, the action space can be represented as a vector:

$$\mathbf{a} = [a_1, a_2, \ldots, a_{30}], \quad \text{where} \quad a_i \in [-1, 1], \quad i = 1, 2, \ldots, 30.$$

Transaction costs are applied to the final cost of each action and are accounted for when calculating the updated balance following a transaction.

D. Reward Function

The reward is computed as the change in portfolio net worth over consecutive time steps. Let w_t represent the agent's net worth at time step t, and let w_{t-1} represent the agent's net worth at the previous time step. The reward function $r(t)$ at time step t is defined as:

$$r_t = w_t - w_{t-1}$$

To discourage risky trading and holding many shares during periods of high market volatility, a penalty is applied when the VIX index exceeds a threshold (VIX $>$ 30). 30 was chosen as the threshold because it is considered to be a marker for extreme swings in the market [8]. If the VIX index exceeds 30 at time step t, the reward is penalized by the total number of shares held:

$$r_t = r_t - \sum_{i=1}^{30} s_i(t),$$

E. Individual Agents

All reinforcement learning agents in this project are based on the Actor-Critic framework, which combines two key components:

Actor: Represents the policy that selects actions (e.g., buy, sell, hold) given the current state. The actor directly updates its policy to maximize the expected return by learning which actions yield the best outcomes.

Critic: Evaluates the actor's actions by estimating the value function, which measures the quality of states or state-action pairs. The critic provides feedback to the actor using metrics such as:

- **State-value** $V(s)$: Expected reward starting from state s.
- **Action-value** $Q(s,a)$: Expected reward starting from state s and taking action a.
- **Advantage** $A(s,a) = Q(s,a) - V(s)$: Quantifies the improvement of an action over the baseline value of the state.

This synergy allows the actor to improve its policy iteratively while utilizing the critic's evaluation to focus on valuable actions, balancing exploration and exploitation. All algorithms were implemented through the `Stable-Baselines3` library [9].

Below is a concise description of each algorithm that was used to train an agent:

- **Proximal Policy Optimization (PPO)**: An on-policy algorithm that ensures stability by limiting policy updates with a clipping mechanism, preventing drastic changes. PPO is well-suited for continuous and discrete action spaces [10].
- **Advantage Actor-Critic (A2C)**: An on-policy algorithm that synchronously updates policies using advantage estimates, balancing bias and variance for efficient learning [11].
- **Deep Deterministic Policy Gradient (DDPG)**: An off-policy algorithm for continuous action spaces that learns

deterministic policies and uses experience replay for stable training [12].

- **Soft Actor-Critic (SAC)**: An off-policy algorithm that maximizes a trade-off between expected returns and policy entropy, encouraging exploration while maintaining stability [13].
- **Twin Delayed Deep Deterministic Policy Gradient (TD3)**: An off-policy extension of DDPG that employs twin Q-networks to reduce overestimation bias and delays updates for improved reliability [14].

These agents leverage the actor-critic framework to tackle the complexities of stock trading, adapting to dynamic market conditions while optimizing portfolio returns.

F. Ensemble Strategies

To optimize decision making, we used ensemble strategies to aggregate the actions of multiple Deep Reinforcement Learning (DRL) agents. The intuition behind this approach is that each agent may perform well under specific conditions but may struggle in others, so an effective ensemble agent would be able to identify and lean on the strongest agent at different times.

Two ensemble strategies are implemented:

- **Weighted Average Ensemble**: This strategy averages the outputs of multiple agents to ensure stability across market scenarios to help mitigate the effects of volatility. The weights assigned to each agent are determined by simulating trading for all agents over a validation period. For agents that generated a positive return, the weight w_i of agent i is computed as follows:

$$w_i = \frac{\text{final portfolio value of agent } i}{\sum_{j=1}^{n} \text{final portfolio value of agent } j},$$

  where n is the total number of agents that generated a positive return. Thus, The weighted average output is given by:

$$a_{\text{WAVG}} = \sum_i w_i a_i, \tag{1}$$

  This approach ensures that agents with better performance (higher portfolio values) contribute more significantly to the final decision. Agents that did not generate a positive return over the validation period are not assigned any weight.
- **Meta-Agent**: This strategy employs a machine learning model, XGBoost, to dynamically predict the best action based on the outputs of the individual agents. The XGBoost model is trained over the same period that the agents were trained over. The input features of the training set consist of all the agent's actions at each day, and the target is the action that generated the highest reward each day.

Ideally, these ensemble strategies address the limitations of individual agents by combining the strengths of multiple agents.

IV. Experiments

The dataset used for testing the agents consisted of daily price data and technical indicators for the stocks in the Dow Jones 30 from 01/30/2018 - 05/08/2022, which covers 1076 trading days. The test period does not start at 01/01/2018 because the first trading days of the month are used to calculate technical indicators. Features include *Open*, *High*, *Low*, *Close*, *Adjusted Close*, *Volume*, and several technical indicators such as *MACD*, *Signal*, *RSI*, *CCI*, and *ADX*. Additionally, daily VIX index data was incorporated to account for market volatility. Further details on the dataset used for testing can be found in section 3.A.

All reinforcement learning (RL) agents were trained using the `Stable-Baselines3` library. Each agent was trained on 50,000 steps, which equates to approximately 50 episodes. The important hyperparameters for each reinforcement learning agent are as follows:

- **PPO**: learning rate = 0.0003, batch size = 64
- **A2C**: learning rate = 0.0007, batch size = 5
- **DDPG**: learning rate = 0.001, batch size = 256
- **SAC**: learning rate = 0.0003, batch size = 256
- **TD3**: learning rate = 0.001, batch size = 256

Ensemble Strategies:

- **Weighted Average Ensemble:** Actions weighted by agent portfolio values over the validation period (01/31/2017 - 12/31/2017).
- **Meta-Agent (XGBoost):** number of trees = 500, learning rate = 0.001, max depth = 6, samples per tree = 0.8

A. Evaluation Metrics and Baseline

Performance evaluation was based on the following metrics:

- **Cumulative Return:** Measures overall profitability.
- **Annual Volatility:** Reflects risk levels in trading strategies on average for each year.
- **Max Drawdown:** Assesses the largest portfolio loss from peak to trough.

These metrics provide a comprehensive view of both returns and risk-adjusted performance. While cumulative return highlights profitability, annual volatility and max drawdown capture the stability of the strategies of each agent.

The Dow Jones Industrial Average (**ˆDJI**) is used as a baseline to benchmark the performance of our DRL agents and ensemble strategies. The ˆDJI index is a price-weighted stock market index comprised of 30 major U.S. companies from various industries – the same companies that our agents are trading on over the test period. The motivation for using the ˆDJI as a baseline is that it contextualizes the returns, volatility, and drawdown metrics of our RL agents' strategies against a more balanced and passive approach to trading on the stocks in the Dow Jones 30.

B. Results and Comparison

Table I summarizes the performance of the individual RL agents and ensemble strategies on the test period, and Fig. 2.

Agent	Cumulative Return	Annual Volatility	Max Drawdown
PPO	-86.68%	0.2026	-87.16%
A2C	75.27%	0.3061	-34.22%
DDPG	63.23%	0.30.51	-34.01%
SAC	-36.60%	0.2581	-52.67%
TD3	63.23%	0.3051	-34.01%
Weighted Average	45.27%	0.2212	-23.39%
Meta-Agent	40.32%	0.1991	-28.14%
Dow Jones Index	25.63%	0.2218	-37.09%

TABLE I

PERFORMANCE OF RL AGENTS AND ENSEMBLE STRATEGIES

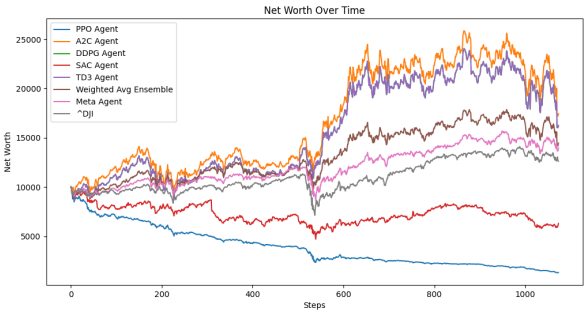

Fig. 2. The net worth for each agent over time for the test period (01/30/2018 - 05/08/2022)

describes the total net worth of each agent from the beginning to the end of the test period.

The output of the Weighted Average ensemble agent is computed as a weighted combination of the individual agents' actions. The weights assigned to each agent are determined based on their portfolio values at the end of the validation period. The computed weights were as follows:

- PPO Agent: $w_{\text{PPO}} = 0$
- A2C Agent: $w_{\text{A2C}} = 0.2526$
- DDPG Agent: $w_{\text{DDPG}} = 0.2555$
- SAC Agent: $w_{\text{SAC}} = 0.2365$
- TD3 Agent: $w_{\text{TD3}} = 0.2555$

The A2C, DDPG, and TD3 agents demonstrated the highest individual returns. The Weighted Average ensemble agent provided more stability as seen by its much lower annual volatility and reduced max drawdown, but it still achieved moderate returns. The Meta agent achieved the lowest annual volatility of all agents, and still achieved a strong return, however, its max drawdown was 5% higher than the max drawdown of the Weighted Average ensemble agent. All of these agents generated higher cumulative returns compared to the baseline ^DJI index. The PPO agent nearly lost all of its initial balance, and the SAC agent generated a poor return as well.

C. Discussion

The SAC agent performed well over the validation period, generating about a 34% return. This positive return resulted in the actions of the SAC agent being assigned a weight of 0.2365 for the Weighted Average ensemble agent. In the test period, the SAC agent performed poorly, so the weight assigned to the SAC agent for the Weighted Average ensemble agent may have hindered the performance of the Weighted Average ensemble agent. This exposes one of the weaknesses of the weighted average approach – an agent generating strong returns over a prior validation period does not guarantee that agent will generate strong returns in the future. To address this issue the weighted ensemble approach could be modified to incorporate a rolling validation window rather than a static one.

It is interesting to observe that the agents that generated the highest cumulative returns also had the highest annual volatility (A2C, DDPG, TD3). Investigating the portfolios of each of those agents revealed that their final portfolios consisted solely of shares of Amazon, and the trend of their net worth over time matches the trend of the Amazon stock over the test period. This suggests that these agents' learned to simply buy and hold what they deemed to be the best individual stock. This is a high-risk, high-reward strategy, as evinced by the test results.

The ensemble agents effectively mitigated risk, as hoped, at the cost of not achieving the highest returns. Their final portfolios were much more diversified, with the Weighted Average ensemble agent and Meta ensemble agent holding shares of 7 stocks and 13 stocks, respectively.

In Fig. 2., the green line, representing the DDPG agent is unobservable because the DDPG agent took identical actions to the TD3 agent and finished with the exact same portfolio (all Amazon stock). We expect the reason for this result is that the TD3 algorithm is an improvement upon the DDPG algorithm, primarily through the utilization of two Q-networks as oppposed to one. Thus, it is not surprising that both algorithms produced similarly performing agents.

D. Conclusion

This project highlights the power of reinforcement learning coupled with ensemble decision strategies in stock trading environments. Ensemble methods effectively balanced returns and volatility, while some individual agents took on more risk and achieved higher returns. Future work could improve ensemble methods by utilizing dynamic validation and incorporating descriptive market features into the input features of a meta agent (i.e. market sentiment analysis). **Github: https://github.com/saivaka/Reinforcement-Learning-Stock-Trading-Bot-**

References

[1] Hongyang Yang et al. "Deep reinforcement learning for automated stock trading: an ensemble strategy". In: *Proceedings of the First ACM International Conference on AI in Finance*. ICAIF '20. New York, New York: Association for Computing Machinery, 2021. ISBN: 9781450375849. DOI: 10.1145/3383455.3422540. URL: https://doi.org/10.1145/3383455.3422540.

[2] Francesco Bertoluzzo and Marco Corazza. "Testing Different Reinforcement Learning Configurations for Financial Trading: Introduction and Applications". In: *Procedia Economics and Finance* 3 (Dec. 2012), pp. 68–77. DOI: 10.1016/S2212-5671(12)00122-0.

[3] Yanjie Song et al. *Ensemble Reinforcement Learning: A Survey*. 2023. arXiv: 2303.02618 [cs.LG]. URL: https://arxiv.org/abs/2303.02618.

[4] Pham The Anh. *Deep Reinforcement Learning for Automated Stock Trading*. Sept. 2024. URL: https://medium.com/funny-ai-quant/deep-reinforcement-learning-for-automated-stock-trading-9d47457707fa.

[5] Taylan Kabbani and Ekrem Duman. "Deep Reinforcement Learning Approach for Trading Automation in the Stock Market". In: *IEEE Access* 10 (2022), pp. 93564–93574. ISSN: 2169-3536. DOI: 10.1109/access.2022.3203697. URL: http://dx.doi.org/10.1109/ACCESS.2022.3203697.

[6] Jie Zou et al. *A Novel Deep Reinforcement Learning Based Automated Stock Trading System Using Cascaded LSTM Networks*. 2023. arXiv: 2212.02721 [q-fin.CP]. URL: https://arxiv.org/abs/2212.02721.

[7] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[8] URL: https://www.td.com/ca/en/investing/direct-investing/articles/understanding-vix.

[9] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[10] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

[11] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning". In: *CoRR* abs/1602.01783 (2016). arXiv: 1602.01783. URL: http://arxiv.org/abs/1602.01783.

[12] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *CoRR* abs/1509.02971 (2015). URL: https://api.semanticscholar.org/CorpusID:16326763.

[13] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *CoRR* abs/1801.01290 (2018). arXiv: 1801.01290. URL: http://arxiv.org/abs/1801.01290.

[14] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *CoRR* abs/1802.09477 (2018). arXiv: 1802.09477. URL: http://arxiv.org/abs/1802.09477.