Application of Deep Reinforcment Learning in Portfolio Management
ECE 570 Course Project

Anonymous submission

Abstract

This study investigates the use of Deep Reinforcement Learning (DRL) for dynamic portfolio management in stock trading, aiming to enhance returns while managing risk. Traditional static models often struggle with the rapid, unpredictable changes in financial markets. By modeling portfolio management as a Markov Decision Process (MDP), we designed an environment where a DRL agent can make trading decisions based on asset technical indicators and covariances. Testing on sixteen years of historical data, three out of four DRL models outperformed a buy-and-hold strategy on the Dow Jones Industrial Average Index, with the A2C model achieving the highest cumulative return and Sharpe ratio, despite its simplicity. These results underscore the potential of DRL for adaptive, high-performing trading strategies, highlighting areas for further exploration in model complexity and reward schemes.

Links expire 01/01/2025
Code — https://anonymous.4open.science/r/Applying-FinRL-Portfolio-Management-7375/experiment.ipynb
Demo Video — https://anonymous.4open.science/r/Applying-FinRL-Portfolio-Management-7375/demo_video.mp4

Introduction

Stock portfolio management involves selecting a mix of financial assets to find a balance between risk and reward potential. It is crucial for investors to strategically allocate their capital among various assets over time based on their investment goals, risk tolerance, and market conditions. Effective portfolio management can help investors achieve long-term financial objectives, hedge against market volatility, and navigate economic uncertainties.

Since the existence of the stock market, analysts have developed models with the intention of creating portfolios that achieve optimal returns while minimizing risk. However, in a highly chaotic, rapidly changing market, static models fall short when context changes. The application of Deep Reinforcement Learning in portfolio management allows for the creation of dynamic models that adapt as market context changes.

Related Work

Algorithmically modeling portfolio management is far from a new idea. However, many financial technology organizations attempt to keep their strategies for portfolio management private because maintain trade secrets allows for organizations to have an edge over others. The existing related work that was studied for this paper was produced by universities studying quantitative finance with the goal of producing tools to allow others to gain 'hands-on' experience to break into the space. A brief analysis of two papers that this implementation contains deep roots from are described below.

Overview of FinRL

This is an application based paper from the 2020 NeurIPS conference that describes an open source framework for applying DRL models in the quantitative finance space. FinRL is broken up into three modular components to apply DRL: environment, agent, and applications. The key contributions of the FinRL paper are listed below, directly quoted as follows:

- FinRL is an open source library specifically designed and implemented for quantitative finance. Trading environments incorporating market frictions are used and provided.
- Trading tasks accompanied by hands-on tutorials with built-in DRL agents are available in a beginner-friendly and reproducible fashion using Jupyter notebook. Customization of trading time steps is feasible.
- FinRL has good scalability, with a broad range of fine-tuned state-of-the-art DRL algorithms. Adjusting the implementations to the rapid changing stock market is well supported.
- Typical use cases are selected and used to establish a benchmark for the quantitative finance community. Standard backtesting and evaluation metrics are also provided for easy and effective performance evaluation. (Liu et al. 2020)

The authors of the paper took a highly modular approach in the implementation of FinRL. Each of the three layers has multiple pre-built modules for implementing custom financial strategies. However, the core components that are used DRL remain the same across the modules that are provided. The trading process is modeled as a Markov Decision Process (MDP) which consists of a State Space, Action Space, and Reward Function. The state space consists of current portfolio and marked data such as user balance, shares owned, market close, market open/high/low price, trading volume, and market technical indicators. The action space allows each individual stock to be bought or sold in number of shares. The reward function can either selected from a handful of options dependant on portfolio value increase, or it can be explicitly defined by the user.

The experiment done in the FinRL paper is a 1.5 year simulation of trading the Dow Jones fund stocks with two unique methods: multi stock trading and portfolio allocation. In both cases, the model beat the market (Liu et al. 2020).

This is a strong paper that showcases the power of DRL in fintech. The structure of FinRL was described in great detail, and the modular nature of the framework was laid out in a straight-forward manor. In terms of application of DRL, this is a fantastic paper for a beginner fintech researcher to 'break the ice' in reinforcement learning. However, portions of the paper are unclear due to the beginner-friendly nature of the paper. For example, there is minimal detail in the specifics on how DRL models are trained. Furthermore, technical details on how the agent learns over time is missing. Multiple gaps also exist for future work. One of the most critical factors in trading is sentiment, and FinRL does not take into account this information. Furthermore, overall market conditions are not explicitly evaluated outside of the portfolio positions that are given to trade.

Overview of DeepTrader

The second related work was a 2021 AAAI conference paper that also took the approach of creating an open-source tool for portfolio management. Novelty in this paper comes from the addition of macro market trends and correlation between assets in the learning model. The key contributions of the DeepTrader paper are listed below, directly quoted as follows:

- We propose DeepTrader, a DRL-based method for riskreturn balanced portfolio management. Market conditions are novelly taken into account as an independent profit-risk balancing module, while the cross-asset interrelationship extraction is enhanced by learning and using a graph structure to characterize interrelationships between stocks.
- Experiments on three stock indexes demonstrate the superiority of DeepTrader in balancing risk and return, especially in the period of subprime mortgage crisis and the recovery period. Ablation studies further confirm the effectiveness of the key components in DeepTrader and the effectiveness of using learned causal structure to encode the interrelationships between assets. (Wang et al. 2021)

Like the FinRL paper, the trading process was modeled as a MDP, but the state, action and reward ar unique. The DeepTrader paper defined the reward function as a combination of both return and risk quantifiers. The action space allows for additionally complex trading operations such as trading on borrowed funds. The state space is the most unique because it does not directly observe market information. Instead, the same market information that FinRL observes is used to create 'scores' for portfolio assets and estimates of overall market conditions which were then used as states. The agent was then tested and trained on three common stock indexes (including the DJIA that was used for the FinRL experiment). The paper demonstrates that DeepTrader outperforms both the market as well as other known trading strategies (Wang et al. 2021).

The DeepTrader paper was a highly complex and effective approach to modeling portfilio management with DRL. Showing that DeepTrader outperforms several existing strategies that already outperform the market was highly effective in showcasing the performance of the model. Furthermore, showing results for multiple portfolios strengthened results.

Comparison and Analysis

When comparing the two previous works, both experiments had the goal of creating an open-source DRL Agent for stock portfolio management. However, FinRL is significantly more generalized when compared to DeepTrader because it has the ability to perform multiple functions outside of portfolio management while DeepTrader was exclusively developed to manage a portfolio.

Both models represent the trading process as a MDP but with variations that underscore their differing objectives. FinRL's state space relies on common technical indicators and covariances, providing direct observations of the market, while DeepTrader's state space incorporates 'scores' based on derived estimates of portfolio and market conditions, rather than raw market data. This abstraction in DeepTrader potentially reduces noise, aligning well with its goal of risk-return balancing. In terms of action space, FinRL allows basic trading actions—buy, sell, or hold. DeepTrader, however, incorporates more complex actions, such as trading with leverage. This aligns with its enhanced reward function, which considers both returns and risk quantifiers, offering a nuanced reflection of market behavior under different risk tolerances.

Both tools were evaluated on stock indexes, with DeepTrader's experiments expanding to three indexes, including the DJIA, which is also used in FinRL. Both tools demonstrated outperformance relative to the market and traditional strategies, but DeepTrader's additional reward complexities and risk-management features provide a unique comparative edge in scenarios that require balancing high returns with controlled risk exposure. When directly compared, the DeepTrader Agent did yield better results than FinRL trading the DJIA, but it was also trained on a larger span of data (Wang et al. 2021).

This brings into question if the reason DeepTrader outperformed FinRL was due to superiority of the model, additional training data, or a combination of both. As a naive undergraduate student with limited knowledge of quantitative finance, it is easy to assume that an agent trained from direct market observations would be superior to one trained on abstracted observations. However, the discrepancies in training data and action spaces leaves a gap between the two works.

Problem Definition

Optimization of portfolio management to balance both risk and return is a highly complex task in a rapidly changing market environment. Specifically, this paper explores the application of Deep Reinforcement Learning (DRL) to create dynamic models that adapt to market fluctuations, overcoming the limitations of traditional static models and human management. The key research question is how DRL can effectively balance risk and return while incorporating real-time market conditions and interrelationships between assets.

Methodology

In the context of stock portfolio management, the process can indeed be viewed as a closed-loop interaction between an **agent** (the portfolio manager or decision-making model) and an **environment** (the stock market). The agent is responsible for making portfolio allocation decisions based on the information it receives about the current market state. It chooses actions such as buying, selling, or holding assets to optimize certain objectives, typically maximizing returns and minimizing risk. In algorithmic trading or automated portfolio management, the agent could be a reinforcement learning model or another algorithm trained to make trading decisions. The stock market, serving as the environment, represents the complex and dynamic financial ecosystem within which the agent operates. The environment provides feedback to the agent in the form of observed changes in stock prices and other market indicators.

Like the previous work, the agent-environment interaction required to manage a stock portfolio was modeled as a Markov Decision Process (MDP). The MDP can be represented below as the following 4-tuple where:

$$MDP = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$$

- $\mathcal{S}$ is a set of all states in the process known as the state space
- $\mathcal{A}$ is a set of actions available to the agent known as the action space
- $\mathcal{P}$ is a function that **deterministically** maps a state action pair (s, a) to a next state s'
- $\mathcal{R}$ is the reward that the agent receives for making the transition $(s, a) \to s'$

Environment

The environment represents the stock market that the agent interacts with and contains the MDP state space, transition function, and reward functions.

State Space:

The state space is the element of the environment that is directly observed by the agent in order to make decisions. The state space for this paper consisted of asset technical indicators and covariances between assets.

Indicator	Description
MACD	Moving average convergence/divergence
Boll-UB	Bollinger upper bound
Boll-LB	Bollinger lower bound
RSI	30 day relative strength index
CCI	30 day commodity channel index
DX	Average directional movement index
SMA-30	30 day simple moving average
SMA-60	60 day simple moving average

Table 1: Technical Indicators

The technical indicators in Table 1 are all commonly used in traditional trading strategies. The Bollinger Band can be used to gauge the volatility of an asset. RSI and CCI are both momentum indicators that can gauge if the asset is overbought or oversold. MACD measures how much a short term average of an asset is approaching the long term average, and the DX measures the strength of the current asset price trend. In addition to the technical indicators in Table 1, the covariances between assets was also used in the state space; this allows for overall market moves to be captured as well as give an indicator of which assets move together. All covariances in this experiment were calculated with a 252 day (one year of trading days) look-back period.

In the experiment, this state space can be represented as a column vector S such that $S \in \mathbb{R}^{N_a(N_a+N_i)}$ where N_a is the number of assets and N_i is the number of technical indicators. Note that the covariance matrix between all assets is of dimension $\mathbb{R}^{N_a \times N_a}$.

Action Space:

The action space represents the trades within the portfolio that should be executed. These are given as a column vector A where $A \in \mathbb{R}^{N_a}$. The elements in an action vector represent the normalized monetary distribution of portfolio funds over the assets. Thus, A is also subject to the constraints:

$$0 \leq a \leq 1 \quad \forall a \in A \qquad \text{and} \qquad |A|_1 = 1$$

In simpler terms, if the first two values in A are 0.10 and 0.20, then the first two assets in the portfolio make up 10% and 20% of the portfolio's value respectively. The same is true for all values in A, and all values add up to

100%. Additionally, one important note here is that this action space does not give the ability to 'hold cash' as an asset.

Transition Function:

Since the portfolio allocations have no affect on stock prices, the transition function here solely was used to observe closing prices of assets for the reward function.

Reward Function:

The reward function calculates the total portfolio value based on the closing prices of each asset that day. The reward R is a single element such that $R \in \mathbb{R}$. The calculation for this experiment follows the below formula:

$$R = A^{\top}(C_t \oslash C_{t-1} - 1)$$

In this case, C_t represents asset closing prices at time t, and $\oslash$ is an element wise division operator.

Agents

A total of four deep reinforcement learning (DRL) agents were trained. This includes the two agents from the original FinRL paper: DDPG and TD3. A very brief overview of each model is given below.

Asynchronous Advantage Actor Critic (A2C):

The A2C model is a synchronous, deterministic variant of a DRL agent developed by Google in 2016. The agent follows a standard Q-learning algorithm estimating the reward function with a Deep Neural Network (DNN) (Mnih et al. 2016). Of the four models, A2C is the both the simplest and oldest, so it was used to provide a general baseline DRL strategy that the more specific ones below should outperform.

Proximal Policy Optimization (PPO):

The PPO model was released by OpenAI to build upon A2C. The main differences between A2C and PPO are that the PPO model can perform multiple updates to a gradient, and a limited number of changes are made to the policy that determines actions. By limiting the number of changes that can be made to the policy in a single step, overestimation bias is able to be reduced (Schulman et al. 2017).

Deep Deterministic Policy Gradient (DDPG):

The DDPG model was also developed by researchers at Google. Instead of using a DNN to estimate the Q-function, DDPG uses a model-free approach to directly estimate the policy that determines actions. Directly estimating the policy allows for improved policy creating in highly complex tasks (Lillicrap et al. 2019).

Twin Delayed DDPG (TD3):

The TD3 model was based on a 2018 ICML article that expands upon DDPG. TD3 limits the improvement of a policy in a time step in order to reduce overestimation bias (Fujimoto, van Hoof, and Meger 2018).

Experiment Data and Setup

The experiment dataset was created by downloading historic stock market data with the YahooFinance API. Historical data for 30 stocks in the Dow Jones Industrial Average was downloaded to represent the assets that could be traded. Stock tickers for each asset are shown in Table 2.

AXP	AMGN	AAPL	BA	CAT
CSCO	CVX	GS	HD	HON
IBM	INTC	JNJ	KO	JPM
MCD	MMM	MRK	MSFT	NKE
PG	TRV	UNH	CRM	VZ
V	WBA	WMT	DIS	DOW

Table 2: Stocks Selected as Assets

Sixteen years of stock data, spanning January 1, 2008, to January 1, 2024, were collected for each ticker listed in Table 2. The dataset was downloaded with a 1-day granularity between the start and end date for all days that the stock market was open. Thus, after hours price of stocks over weekends and holidays was not included. For each trading day, the dataset provides: stock volume, opening price, daily high and low prices, and the closing price. The technical indicators and covariances between each asset were calculated using this information.

Since stock trading has a heavy dependence on the temporal element of data, the dataset could not be randomized or broken into batches for training. For all models, the first fifteen years was used to train the agent, and the final year was used for testing.

Experimental Results

Once each of the models was trained, the final year of the sixteen year dataset was used for testing. The cumulative return of each model trading the DJIA compared to the performance of buying and holding the index are depicted below in Figures 1 - 4.

In addition to the figures displaying returns, the final cumulative return, sharpe ration, and max drawdown for each model is displayed in Table 3. Sharpe Ratio gives insight into the quality of those returns by assessing the risk-adjusted performance, indicating whether the returns are worth the volatility. Max Drawdown reflects the worst observed loss from peak to trough, highlighting the model's downside risk and resilience during downturns. Together, the Sharpe ratio, max drawdown, and cumulative return together provide a more wholistic picture of each model's performance.

Model Name	Base	A2C	PPO	DDPG	TD3
Total Return	13.2%	**14.7%**	12.8%	14.1%	14.0%
Sharpe Ratio	1.226	**1.255**	1.098	1.186	1.146
Max Down	**-8.6%**	-9.1%	-9.1%	-10.1%	-9.9%

Table 3: Model Performance

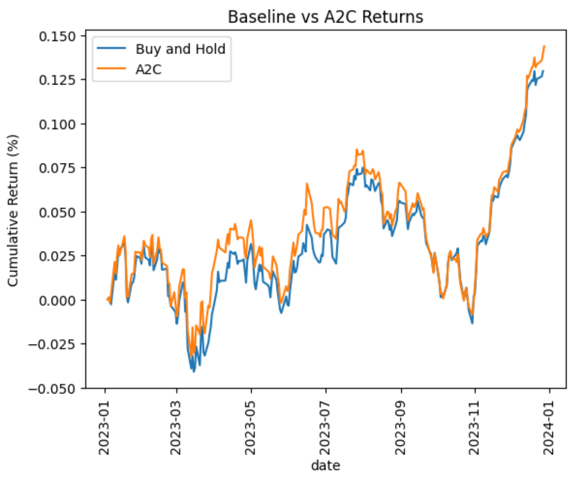

Figure 1: A2C Returns

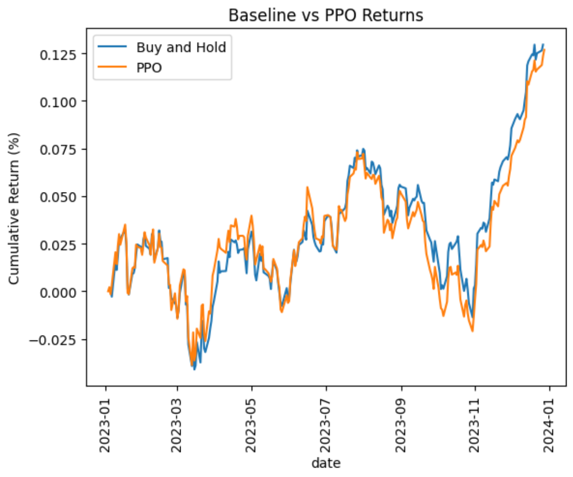

Figure 2: PPO Returns

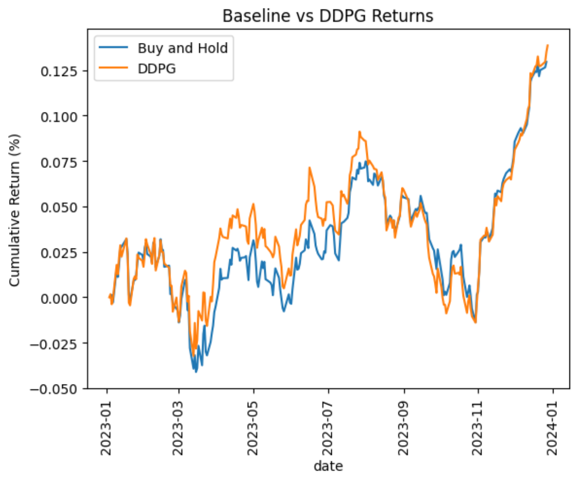

Figure 3: DDPG Returns

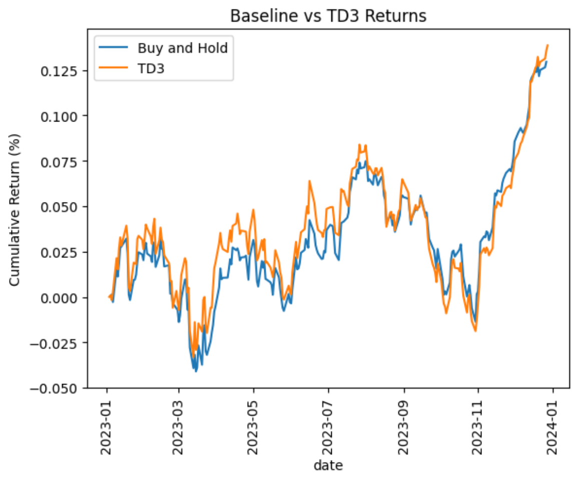

Figure 4: TD3 Returns

The test results for each model show that applying DRL Agents to a trading environment is able to outperform the Dow Jones Industrial Average. While not every model outperformed the index, three of the four did. Based on the final cumulative metrics in Table 3, the A2C model yielded the best cumulative return and Sharpe ratio. This was unexpected because the A2C model was the most simple of the DRL agents that were trained, and it was meant to serve as a lower performing baseline for the other three agents.

While A2C had produced the best final results, a visual inspection of Figure 3 reveals that the cumulative returns for the DDPG agent were on average higher over the duration of the testing period. The DDPG model reduces model complexity by using a DNN to directly estimate trading actions. Higher performance of the DDPG agent was more in line with expectations because of the complexity of the state space and highly stochastic nature of the stock market.

Another notable result is that the PPO agent and TD3 agent both performed worse than their respective A2C and DDPG counterparts. Since both of these models had features that prevented policy overestimation, it is possible that throttling policy changes caused the agents to not learn an effective trading policy over the training period.

Conclusion and Future Directions

This project demonstrates that deep reinforcement learning (DRL) agents can effectively outperform the Dow Jones Industrial Average (DJIA) in a trading environment, particularly when assessing cumulative return, Sharpe ratio, and maximum drawdown metrics. Three of the four DRL models exceeded the returns of a traditional buy-and-hold strategy, highlighting the potential of these algorithms in high-stakes, dynamic environments like stock trading. Notably, the A2C model outperformed other DRL agents in terms of both cumulative return and risk-adjusted return, even though it was initially considered a simpler, baseline model. This unexpected result suggests that in certain environments, simpler DRL architectures may yield strong performance without the additional complexity of advanced models.

The DDPG agent, though it did not achieve the highest cumulative return, demonstrated consistent returns throughout the testing period, which aligns well with its design focused on optimizing actions in continuous state spaces. Meanwhile, the PPO and TD3 agents underperformed relative to their simpler counterparts, potentially due to their policy-throttling mechanisms that may have limited their ability to adapt effectively to stock market fluctuations.

Based on these findings, there are still several possible directions for future work.

1. Implement a cash system. Normally a portfolio manager has the ability to hold cash when anticipating a stock to go down. The current environment is constrained to having all funds fully invested in the portfolio's assets, so there is currently no way to for the agent to 'hold cash.'

2. The environment can still further be fine-tuned, and off policy examples can be explored. The reward scheme that was used for this experiment was based on the total portfolio value. If the goal is to outperform an index, a reward scheme of the difference between the agent and index could create an off-policy that can be transferred to the original model.

3. Increase the robustness and size of datasets, and perform additional preprocessing. Ideally, the goal of any portfolio is to trade only 'good' assets. Selecting stocks from the DJIA held the assumption that they are 'good' assets. Trading on a larger portfolio such as the S&P500 index, would add an additional factor of figuring out which assets are 'good' before deciding to trade them.

References

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. arXiv:1802.09477.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2019. Continuous control with deep reinforcement learning. arXiv:1509.02971.

Liu, X.-Y.; Yang, H.; Chen, Q.; Zhang, R.; Yang, L.; Xiao, B.; and Wang, C. D. 2020. FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance. *Deep RL Workshop, NeurIPS 2020*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Wang, Z.; Huang, B.; Tu, S.; Zhang, K.; and Xu, L. 2021. DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1): 643–650.