SMU Data Science Review

Volume 5 | Number 2 Article 7

2021

Intelligent Investment Portfolio Management using Time-Series Analytics and Deep Reinforcement Learning

Sachin Chavan
SMU, sachavan@mail.smu.edu

Pradeep Kumar
SMU, ppradeepkumar@mail.smu.edu

Tom Gianelle
SMU, tom.gianelle@gmail.com

Intelligent Investment Portfolio Management using Time-Series Analytics and Deep Reinforcement Learning

By Sachin Chavan, Tom Gianelle, Pradeep Kumar
Advisors: David Stroud and Dr. John Santerre
Master of Science in Data Science, Southern Methodist University,
Dallas, TX 75275 USA

Abstract. – With globalization, the capital markets have exploded in size and value, making them exceedingly difficult to predict. These days the public has access to real-time data of the market-leading to more participation. As a positive step, this might lead to better wealth distribution in the society, and it also adds to the random nature of the market, making it more unpredictable. The portfolio accounts consisting of stocks and bonds are considered serious investment assets. They can make or break a person's future. It is also a way of shielding one against market risk or rising inflation. These accounts, when managed well, will grow in value and can keep an individual's finances healthy. The management of these accounts requires good knowledge of markets and periodic adjustment of assets in the account.

1 Introduction

The reality for most individuals is that investing in the public market exchanges is the only one of very few methods to generate wealth. Typically, individuals with limited assets are not afforded the same investment guidance level as those with wealth managers providing investment advice. There are a growing number of AI/ML automated portfolio and investment advisor tools, and the goal of this paper is to leverage the success of the AI/ML stock portfolio trade management research to date. This paper will focus on use of existing Reinforcement Learning (RL) algorithms (Q-Learning & Policy Optimization) and extend the RL methods with time-series analytic techniques. As the stock market evolves, the time-series analytic techniques are meant to ensure the RL environment can effectively improve the probability that the RL agent is making useful stock trading decisions [9].

Financial trading companies are famously known for referencing in their advertising that "prior market results are no guarantee of future performance success." The strength of Reinforcement Learning is RL agents can learn and adjust their behavior while interacting with a live environment. It generates its own training data after every interaction and either receives a reward or a penalty. It interacts with the environment to maximize cumulative rewards. In the portfolio management case, the agent interacts with the environment with a buy, sell or hold action for stock or EFT which is part of the account. The stock trading environment is an uncertain environment with exceptionally large state space. The agent making decisions in such an uncertain environment is modeled with Markov Decision process (MDP). MDP provides a mathematical framework for sequential decision making under uncertainty. Given that financial markets are stochastic, sample parameters that may influence markets are supply, demand, political stability, war, weather, holidays, age/ethnicity/culture of population, pandemic, etc. RL effectively utilizes MDP with several policy iterations to determine effective strategy to interact in a stochastic environment. The stochastic environment consists of the 30 stocks listed in the Dow Jones stock exchange index in this paper's case.

Current work on Reinforcement Learning (RL) investment portfolio management focuses on comparing different RL algorithm approaches. Traditional deep learning methods using various Policy Optimization RL algorithms were employed in prior studies [1,2, 22]. The results of prior studies showed evidence of positive outcomes in comparison to respective baseline trading portfolio techniques (e.g., momentum trading, cost average trading). Like prior referenced research, PPO, A2C, DDQN are samples of RL Deep Q-Learning algorithms that will be studied as part of this paper. The list is not exhaustive but is meant to be illustrative of the options for building RL models that produce useful comparative stock trading model-free policies [3,4].

Remaining consistent with objective of optimizing trades for everyday investors, the selected Deep Reinforcement Learning algorithms will be optimized for analytical comparison of three typical investment options (1) Dow 30 multi-stock trading (2) Dow Electronic Trading Fund (ETF) single stock trading (3) Dow 30 portfolio rebalancing trading. These three investments are considered "typical" because they are the most likely to be found in some form in an individuals' trading account or 401-K. The research will factor in real-world trade costs that identify trading models with lower overall trading costs that have evidence of outperforming the baseline model. The baseline model will consist of straight Dow stock cost-averaging trades, at regular intervals, over the equivalent time period of paper's study.

2 Related Work

Managing assets in portfolio is challenging and daunting task. The primary job of any portfolio manager is to minimize risk while maximizing returns. The portfolio optimization is multi-objective optimization problem and one of the most critical issue in the finance domain. The methods of managing and optimizing portfolios have been evolving and becoming more sophisticated with technology. In last few years, several research papers have been published demonstrating diverse ways of optimizing portfolio and recently deep learning and reinforcement techniques are becoming increasingly popular on the top of traditional methods. The modern portfolio theory was proposed by Harry Markowitz (1952), in a paper published in Journal of Finance. This work later earned author a Nobel prize. The model proposed by Markowitz is also known as Mean-Variance model. The model is primarily used to make decision buy, sell, or hold. However, as far as portfolio optimization is concerned it does not meet expectations of today's financial world.

The most recent paper published by Hieu, L. T. et al. (2020) has experimented with stock portfolio optimization by applying continuous action space reinforcement learning techniques. It presented minimum variance theory for stock subset selection and wavelet transform was used to extract multi-frequency data pattern. The model takes transaction and risk factor into account and RL agent was trained using Deep Reinforcement Learning Algorithms such as Deep Deterministic Policy Gradient (DDPG), Generalized Deterministic Policy Gradient (GDPG) and Proximal Policy Optimization (PPO) [7]. By just utilizing close price, high price and close price after wavelet transform and experimental result has shown DDPG performed better than others.

Soleymani & Paquet (2020) have presented another unique portfolio optimization approach that combines automatic feature selection and policy learning [16]. It employs online and offline learning. The concept is called DeepBreath [16]. The stock transactions sometimes take time to finalize in real-time and it is called settlement risk. The paper has shown the way to tackle settlement risk by using blockchains. As described in the paper restricted stacked autoencoders are used for dimensionality reduction and feature selection and convolutional neural network was used to learn and enforce policy. Input to autoencoder is stock prices and technical indicators derived from prices. Autoencoders' strength is that they can learn non-linear patterns in the data, perform dimensionality reduction, and eliminate correlations. It is an encoder/decoder architecture where encoder reduces dimensions and decoder reconstructs input data from the hidden layers. It is called restricted because decoder partially reconstructs input vector from the hidden layers and shown with experiments that this approach proven to be effective. The features learned by autoencoder are passed to CNN which learns and implement policies using deep reinforcement learning framework. Soleymani & Paquet (2020) has shown that CNN learns only from historical data with SARSA (state-action-reward-state-action) which is called offline learning and for online learning online stochastic batching technique was used.

In another research, Aboussalah & Lee (2020) used Stacked Deep Dynamic Recurrent Reinforcement Learning (SDDRRL) for building optimal portfolio and model built using this approach found to performed superior to Mean-variance optimization, risk-parity model and uniform buy-and-hold (UBAH) index [17]. Typical Recurrent Network has limitation to react to new market conditions. To handle this the paper has introduced concept called Time Recurrent Decomposition.

In one of the papers by Wu, X. et al. (2020) have proposed idea of using quantitative trading approach by using Deep Reinforcement Learning over intuitional approach. The intuitional approach is a typical gut-feeling-based approach of trading and prone to major losses caused by investors' irrational behavior. It proposes mining historical stock data to maximize profits volatile market conditions. It has demonstrated using Gated Recurrent Unit, it is possible to extract a pattern that is responsible for the movement of the index, which can be effectively used along with Deep Reinforcement Learning to maximize profits [9]. It has proposed two adaptive trading strategies GDQN (Gated Deep Q-learning trading strategy) and GDPG (Gated Deterministic Policy Gradient trading strategy) and both strategies have been tested in volatile stock markets and compared with Turtle trading strategy and results shows both RL strategies outperforms Turtle strategy and out of two RL strategies GDPG found to be more stable [9].

In another interesting paper by Wang et al. (2020) employed mixed method using Long short-term memory networks (LSTM) for stock prediction and Mean-Variance model for portfolio optimization [10]. The paper shows experiment was carried out and results were compared between LSTM, Support Vector Machine (SVM), random forest, Deep Neural Networks and Auto regressive Integrated Moving average model (ARIMA) in first stage and Mean-Variance Portfolio model was applied in second stage to optimize portfolio. It concluded that mixed method outperforms others in terms of cumulative returns per year while minimizing risks.

Literature on using Q-Learning approaches explored the issues of designing a multiagent system that aims to provide an effective decision support for daily stock trading problem. One example of note is the MQ-Trader model. MQ-Trader defines multiple Q-learning agents in order to effectively divide and conquer the stock trading problem in an integrated environment [6]. In MQ-Trader the signal agent is provided with following 10 technical indicators that include the relative strength index, MA convergence and divergence, price channel breakout, stochastics, on-balance volume, MA crossover, momentum oscillator, and commodity channel index. There is interest in the use of relative strength indicator and moving average indicators as these overlap with other noted in reference papers [1, 2, 5]. The results were that the learning framework produces better trading performances than the systems based on other alternative frameworks. However, there was no specific comparative examples of the alternative frameworks.

Mnih et al. (2016) explores the need for using asynchronous methods building a Reinforcement Learning system. Multiple policy learn methods are to be available and the use of them varies depending upon the state of an environment. If it is large and using Q-Tables is infeasible, Deep NN are used as function approximators for the state space. These Deep NN are also referred to as Deep Q networks since the output of these Deep NN are Q-values for all the actions that are to be taken. The data to train this Deep Q-Network is collected after random actions are taken by agent. The next state and rewards after taking this random action will be stored in experience replay memory. The random samples from experience replay memory will be taken to avoid correlation between sequential samples. The approach taken by Mnih et al (20160 is worth considering is to create multiple environments in parallel which will explore the environment using different epsilon greedy strategy so that each individual environment is uncorrelated to each other.

Li et al. (2020) explores the rich features of a framework to implement Reinforcement learning system. The applications of Deep Neural networks are numerous. They are used in Health care, Finance, Automotive industries to name a few. As the data sets become larger, the need to distribute the training across multiple compute resources become more prominent. Framework analysis is key to being able to establish a model-free policy capable of providing predictive capability. As noted in this research PyTorch provides a data parallel model which will facilitate the training of Neural networks in parallel. During the training process of DNNs, the forward pass, backward pass and weights optimizations is done in a loop. This loop count will get larger as the size of DNNs and data increases. Using data parallel package, the applications can create multiple replicas of models using a portion of the available data.

These models can synchronize the gradients and weights to achieve a faster training. The data parallel package provides convenient APIs to achieve this task. The data parallel APIs are designed to be non-intrusive. The application developer does not have to rewrite large part of the code in case they find local resource to be limiting and want to distribute the workload across multiple resources.

Chong, T., Tai-Leung, N., Liew, W., & Khim-Sen, V provide useful guidance of use of relative strength indicator (RSI) and moving average convergence-divergence (MACD) indicator as features that can address challenges encountered with stock market volatility. The challenge is that in small time steps stock processes can at times vary widely. It is only when use of RSI and MACD can inflection points can be detected. The inflection points of when a stock is trending up or trending down. The paper revisits the performance of the two trading rules in the stock markets of five other OECD countries [5]. The authors provided evidence that the MACD and RSI indicators consistently generate significant abnormal returns in the Milan Comit General and the S&P/TSX Composite Index. In addition, RSI rule is also profitable in the Dow Jones Industrials index. Key observation point is the research referenced profits are sustainable in the presence of a 1% round-trip transaction cost. The conclusion results provide evidence supporting investors believe in these two technical indicators in developed markets.

Prior work reported that RL modeling is one of the state-of-the-art techniques machine learning techniques among others that in recent years are providing a risk-adjusted return superior to the S&P 500. The reason noted is that the traditional statistical learning algorithms cannot cope with the non-stationary and non-linearity of the stock markets [1,2]. Both evaluate related variants of Deep Q-learning, double DQN and dueling double DQN [2]. Baseline comparison RNN-LSTM model with a greedy strategy approach was used. The greedy strategy means buying every-time the stock is predicted to go up and selling if the stock is predicted to go down. Important observations are (1) the state needs to contain as much information of the factors that affect stock prices as possible. (2) The state needs to contain less noise so the agent can learn right experience and is more likely to choose right action at each time step so that a positive long-term accumulative reward can be acquired [1]. The focus is on having the right information and reduce noise or volatility. Chen et al. (2019) provided example that Deep Q-learning is Q-learning with the Q-table be replaced with a deep neural network. The paper presents use of RNN-LSTM as replacement. The conclusion deep Q-network can learn profitable patterns from raw stock trading data and utilize them to achieve high accumulated reward.

W. Muller and H. Schumann EDA of time-series points to analytical uses of converting data into visualizations [20]. Silva and Catarci show innovative techniques focus on helping users identify periodic patterns in the data [21]. These techniques focus on the issue of optimal space management, enabling the display of an increased number of time-series compared to a single temporal representation. The visualizations have shown to be effectively processed in CNN models to provide create useful RL policies that set framework for RL environment state, actions, and rewards.

3 Data

The initial exploratory data analysis (EDA) helps to establish a RL environment so that RL agent learns to reach goal as efficiently as possible. The agent interacts with an environment to trade on stocks and receives positive reward for trading gain or receives negative reward for trading loss. The detailed EDA shall construct a trading environment so that agent can maximize cumulative rewards with minimum number of interactions or minimum number of trading.

3.1 Time-Series Data Analysis

In the EDA preprocessing analysis this research was able to leverage readily available stock data from Yahoo Finance and Wharton Data Services. There was no need to impute data or address any missing data. The preprocess data step leverages the features selected from the exploratory data analysis. As part of research the exploratory data analysis focus on identifying features that improve the predictive accuracy for stock returns. In practice investment trading, various information needs to be considered, for example the historical stock prices, current holding shares, stock technical indicators, macroeconomic and micro economic data are used for stock prediction. Stock historical prices alone are not sufficient to create a useful prediction model. As shown in Fig 1, the historical stock prices have slow dampening autocorrelations. The slow dampening autocorrelations provide evidence that the historical data does not have strong correlation between time lags and not useful for prediction independently.

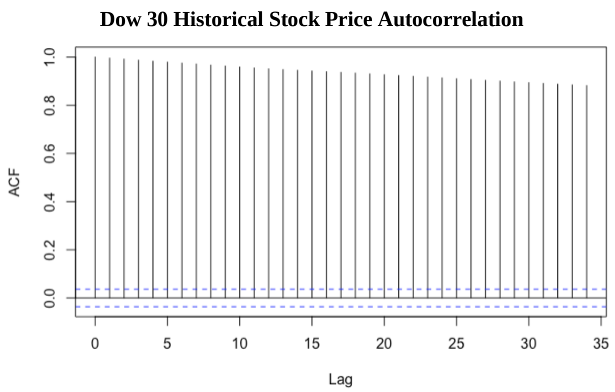


Fig 1: Dow 30 (ETF) Historical Stock Price Autocorrelation (ACF vs Lag)

The time-series data analysis involves use of the “stockstats” package. The “stockstats” features are calculated for feature selection purposes. This paper study included EDA of “stockstats” technical indicators (Moving Average, Relative Strength Indicators, etc) to identify features that would be useful in establishing a Reinforcement

Learning Environment and allow the algorithms to experiment to learn when to trade an investment vehicle (e.g., Dow stock).

In Fig 2 the technical indicators are illustrative of the initial RL Environment State values. As part of the research feature engineering was conducted to assess evidence of correlation between the stock market closing price (both trend and detrend) and "stockstats" technical indicators. In addition, we experiment with a traditional simply move average (SMA) 20-Day vs 200-day crossover indicator created as part of the study. The feature engineering included time-series analysis using vector autoregression (VAR), Pearson correlation coefficient and rolling 30-day correlation of technical indicators vs the Dow Jones Index (Ticker: ^DJI) closing stock price.

In Fig 2, the Lead/Lag charts are used to visually show the output of the rolling 30-day correlation (technical indicator vs DJI closing stock price). The black dotted line indicates the center where an indicator either leads or lags the closing stock price. The red dotted line shows optimal correlation and the set number of days the indicator leads or lags. Numerous technical indicators were researched but the paper only shows results in Fig 2. Only indicators that lag stock close price are useful for signaling whether to "buy", "sell" or "hold" a stock. Based upon research it showed evidence that MACD, CCI and RSI may be use for signaling stock trade decision predictions. 20-day vs 200-day crossover and ADX showed evidence of not being useful for signaling stock trade decisions.

As will be outlined in Methods Section below, the selected technical indicators will be set as the RL Environment State for the RL policy agent neural network (MLP and LSTM) to determine actions that will optimize the RL reward (i.e., portfolio account value).

NOTE: The correlation is not meant to imply causation relationship between technical indicator and the DJI closing stock price. The technical indicators are used by RL policy agent to assist with signaling whether to "Buy", "Sell", or "Hold" a stock. In Fig 2 Technical Indicator definitions source: Investopedia.com.

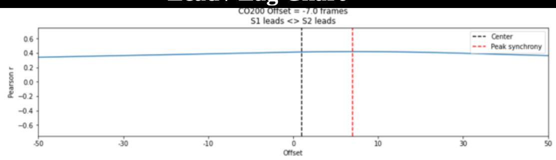
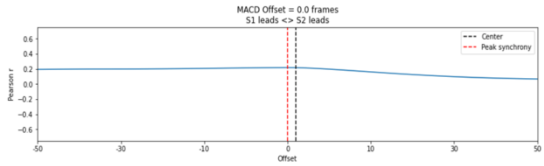
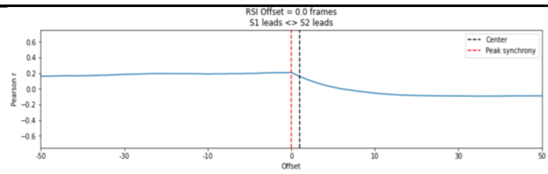
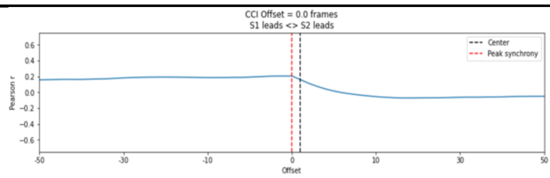
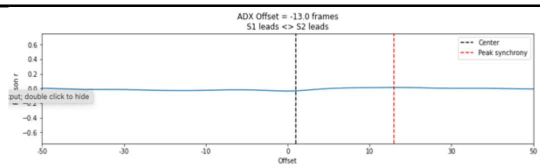
Lead / Lag Chart	Corr
 20-Day SMA vs 200-Day SMA Crossover	Leads
 **MACD** - Moving Average Convergence Divergence	Lags
 **RSI** - Relative Strength Index	Lags
 **CCI** - The Commodity Channel Index	Lags
 **ADX** – Average Directional Index	Leads

Fig 2: Technical Indicators vs Dow Jones Index (ETF)

A complete list of technical indicators is available at the stockstats documentations site. Macro-economic data such as US Treasury Bonds, Commodity Prices (e.g., Copper, Lumber, Gold) and the use of a pre-defined “turbulence” indicator were investigated. The same time-series EDA steps conducted for "stockstats" was also applied to other potential useful predictor data.

It is worth noting that as part of the study random data was created to simulate white noise and used with the RL algorithm to help gauge the validity of the policy predictions vs the use of lagged time-series financial stats and macro-economic data. This white noise test provided a validation check that the data identified as part of feature engineering is useful. This test helped provide evidence that the technical indicators acted as useful features for improving the RL cumulative reward value.

The feature engineering process utilized in this study allows for the RL Environment state to be established at regular intervals. Features that may prove useful in a trending market may not be as useful as a volatile market. The time-interval for this research was kept to a quarterly time-period.

4 Methodology

For research, this paper leveraged the Reinforcement Learning (RL) stock trading framework from "*AI4Finance LLC*". Below paper outlines the components of RL and how research was conducted with various RL Environment state spaces [22].

A reinforcement learning system was designed to find effective investment strategy to either to trade a single stock or multiple stocks or to build a diversified equity focused portfolio with periodic automatic rebalancing to achieve goal of maximizing profits with minimum risks. Reinforcement learning is a novel approach in the field of portfolio management and lot of research is currently being done in this area. These methods are proven to be useful in stochastic environments such as trading environment on the public market exchanges where decisions must be taken under uncertainty.

The challenging part of building Reinforcement Learning system is to build an environment which can mimic the problem and its eco system at hand. As was described in the Section 4.2.3 below, the environment is made up of the stock portfolio account and stock analytical technical indicators. The stock portfolio account changes with every action taken by the agent. Even though the action is generated outside of the environment, the design of action space is tied to overall functioning of the environment. The design of environment state space, action space, reward function and policy agent are below in Sections 4.2 and 4.3.

4.1 Overall Reinforcement Learning (RL) Performance Comparison Approach

In this section, the performance comparison results are shown. This research included performing back-testing for the individual agents (e.g., A2C, PPO, DDPG) and the ensemble approach. Following three stock trade scenarios are modeled to identify stock trading strategies for producing better Sharpe Ratios relative to baseline Dow Jones Index (^DJI) performance.

1. Dow ETF Single Stock Trading
2. Dow 30 Multi-Stock Trading
3. Dow 30 Portfolio Rebalance

This paper started with Dow ETF (Electronic Trading Fund) single stock trading in order to research the "stockstats" technical indicators and other methods (e.g., time-series lag analysis). The Dow ETF single stock trading modeling allow for quick analysis on whether an indicator showed evidence of being useful for signaling whether to "buy", "sell" or "hold" stocks. If technical indicator showed evidence of being useful, then it was used for multi-stock and portfolio rebalance modeling.

To produce comparative results to Dow Jones Index (^DJI) performance this paper leveraged a financial industry standard practice of "backtesting" that allowed for establishing both Sharpe Ratio and comparative charts of study scenario vs ^DJI baseline. The overall Sharpe Ratio for Dow Jones Index Jan 2009 – July 2021 is ~0.74. For this paper the Sharpe Ratio of ~0.74 was the selected baseline comparison for establishing the RL Environment State.

4.1.1 Sharpe Ratio – Overview

Sharpe Ratio is used for general comparison of RL algorithm model return of investment trades vs portfolio risk. Sharpe Ratio is widely used in the Finance Industry to determine strength of portfolio results. The ratio is the average return earned more than the risk-free rate per unit of volatility or total risk. Volatility is a measure of the price fluctuations of an asset or portfolio. For this paper, the yield for a U.S. Treasury bond is used as the risk-free rate. Below is the calculation of the Sharpe Ratio:

$$\text{Sharpe ratio} = (Rp - Rf) / \sigma p \quad (xx)$$

Where:

Rp =return of portfolio
Rf =risk free rate
σp =standard deviation of the portfolio's excess return

Overall, the greater the value of the Sharpe ratio, the more attractive the risk-adjusted return. As a comparison prior to Covid pandemic the Dow ETF (^DJI) had a Sharpe ratio of 0.90. After the Covid pandemic, the Dow ETF (^DJI) Sharpe Ratio for past 3 years has been lower at 0.74. According to "Seeking Alpha" website, a good financial advisor (CFA) typically has a Sharpe Ratio of 0.75 - 1.00. Above 1.00 Sharpe Ratio is considered particularly good and above 2.00 is exceptional.

4.2 Reinforcement Learning (RL) Environment Components

4.2.1 RL Environment Component Overview

Reinforcement learning system consist of different components as depicted in Fig 3. The first component in this is the portfolio account which is made up of the investment funds and count of stocks that are part of the account. The second component is the state of the market which affect the net worth of the account and it changes with every tick of time. The third component is the intelligent agent which will observe the current state of the market and take actions to optimize the portfolio account. A stochastic model to represent the state of the market needs to be built. Markov Decision Process (MDP) is chosen as a modeling method to represent the market Markov model assumes that the current state of the market is a good indicator to predict the future state provided the current state includes sufficient statistic to represent the history of the environment. To build a state to satisfy this requirement, various technical indicators of the stocks are included into state. These technical indicators like relative strength index (RSI) are calculated over a period called look back period providing a good representation of the stock for the look back period.

The agent is supposed to take a decision based on current market conditions and state of the portfolio account. A neural network is considered as an agent.

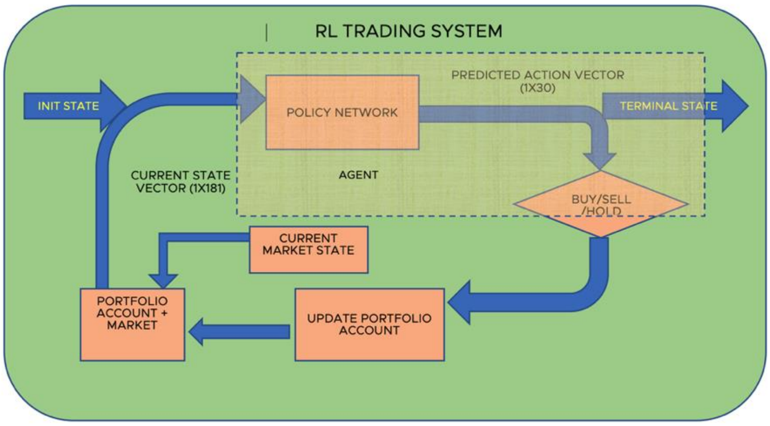


Fig 3: Block diagram of Reinforcement Learning System

The block diagram above depicts the reinforcement system for portfolio account management. The Agent consists of "policy network" and module to execute the predicted action. The input to policy network is the combination of current state of the market and current state of the portfolio account. Based on this input, policy network will predict a vector of action. These actions will be applied on the account to change the net worth of the account.

4.2.2 RL Environment Component - Action Space

Single and Multiple Stock Trading

In the case of single stock trading and multiple stock trading environments the possible actions in the trading environment are Buy, Sell or Hold action for any stock in the account and the number of stocks that needs to be acted upon. The job of the agent at any step is to predict the right set of actions on x number of stocks to maximize the rewards. The experimental portfolio account is limited to 30 stocks in the DOW index. The action space at any tick of time needs to generate suggestions to act upon all 30 stocks which makes it a vector of size 30. Each individual values in this vector are decimals between –1 and +1. The interpretation of these decimal values by the environment can be explained with example below.

[AAPL, HD, INTC, IBM, …........] = [-0.9, 0, 0.8, -0.1, …...]

Consider the vector above which shows only first 4 values of a vector which is of the size 30. This vector is associated with individual stocks by its position. The value –0.9 is an action to be taken on AAPL stock. The environment normalizes this action by multiplying with 100 before interpreting and acting. The value –0.9 will become –90 after normalization and the negative value indicates sell 90 stocks of AAPL which are in the portfolio account.

Portfolio Rebalancing

The stock markets are dynamic, volatile and situation changes every day. The stocks which were profitable yesterday become riskier today and vice versa. In diversified portfolio riskier stocks may hurt overall portfolio standing. The job of portfolio manager is to watch market closely and take decisions to buy and sell stocks periodically to maximize returns. The portfolio rebalancing environment achieves the same. The difference between multiple stock trading and portfolio rebalancing is, in multiple stock trading RL strategy is applied to each stock independently that maximizes returns on that specific stock, but portfolio rebalancing maximizes returns on overall portfolio.

In this environment, the reinforcement learning agent learns to reallocate weights of stocks to maximize returns. These selected weights are positive numbers between 0 and 1 such that sum of all weights for stocks in portfolio is equal to 1. The agent returns weight vector as an action vector. The length of this action vector is equal to total number of stocks in the portfolio. For this paper portfolio account is limited to 30 stocks in the DOW index therefore action vector is of length 30. Each element in the weight vector represents a proportion of stock on the day of trading. This environment was setup for daily rebalancing, so reallocation is done every day at the end of trading session. The sample action vector is as below.

[AAPL, HD, INTC, IBM, …........] = [0.1, 0.0, 0.3, 0.2, …...]

date	AAPL	AXP	BA	CAT	CSCO	CVX	DD	DIS	GS	HD	IBM	INTC	JNJ	JPM	KO	MCD
1/2/19	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333	0.03333333
1/3/19	0.02376958	0.06461241	0.02376958	0.02376958	0.02376958	0.02376958	0.02376958	0.02376958	0.02376958	0.06461241	0.02376958	0.02376958	0.02376958	0.02376958	0.02376958	0.05769359
1/4/19	0.02873329	0.0225386	0.06126627	0.06126627	0.02408434	0.0225386	0.0225386	0.03430975	0.02311797	0.0225386	0.02447389	0.06126627	0.02283042	0.0225386	0.0225386	0.0225386
1/7/19	0.02153372	0.02153372	0.02153372	0.05718386	0.05853472	0.05853472	0.02623886	0.05853472	0.02153372	0.04317012	0.02153372	0.05853472	0.05606836	0.02790722	0.02153372	0.02153372
1/8/19	0.01874963	0.01874963	0.01874963	0.01892596	0.033539	0.05096679	0.01874963	0.02552116	0.05096679	0.0455865	0.01874963	0.01874963	0.05096679	0.01874963	0.01874963	0.04644862
1/9/19	0.0239742	0.0239742	0.0239742	0.0239742	0.0239742	0.06238881	0.06516863	0.0239742	0.06516863	0.0239742	0.0239742	0.0239742	0.0378221	0.02662954	0.02658862	0.0239742
1/10/19	0.05799437	0.02133493	0.02133493	0.04637349	0.04530002	0.02683064	0.02133493	0.02133493	0.03850462	0.03327378	0.02133493	0.03573659	0.02133493	0.05799437	0.02690285	0.02133493
1/11/19	0.06495952	0.02389727	0.02389727	0.02389727	0.02389727	0.0468875	0.02389727	0.06495952	0.02389727	0.02620144	0.06495952	0.06495952	0.02389727	0.02389727	0.0339397	0.04436569
1/14/19	0.0250906	0.0250906	0.0250906	0.0250906	0.02659444	0.0250906	0.0250906	0.06820332	0.04649722	0.06820332	0.0250906	0.03955504	0.06820332	0.02946702	0.0250906	0.0250906
1/15/19	0.03348049	0.02473774	0.02473774	0.02473774	0.06724415	0.02473774	0.03162117	0.02473774	0.0294669	0.02473774	0.06724415	0.03104524	0.02529501	0.03688934	0.02473774	0.06724415
1/16/19	0.04731941	0.0429178	0.02393715	0.02614461	0.02393715	0.02393715	0.02393715	0.05367145	0.02393715	0.02393715	0.06231197	0.02393715	0.02393715	0.02860939	0.02393715	0.06506791
1/17/19	0.02121659	0.02121659	0.03906495	0.0244201	0.02121659	0.02121659	0.03072657	0.02644635	0.02121659	0.03912799	0.02208512	0.04580799	0.05767267	0.02121659	0.05767267	0.02121659

Fig 4: Sample actions taken by agent

Sample actions in Fig 4 does not show actions on all 30 stocks due to space limitations. It shows actions taken on the first 16 stocks from day 1 to 12 consecutive days. On day 1 all 30 stocks in portfolio carry same weight and sum of all weights on each row is 1. The rebalancing moves money out of risky asset to assets with lower risk thus minimizes overall risk of the portfolio.

4.2.3 RL Environment Component - State Space

Single and Multi-Stock Trading

The Reinforcement Learning agent makes a decision about its next action based on current market conditions and the current state of the portfolio account. The RL agent needs these two data to define the state space or state vector. State space vector combines the data of current market conditions and current state of the portfolio account. In single stock and multiple stock trading environments dimension of state space vector is 181x1. The contents of the vector are as shown below.

[‘Available funds/Balance], [Adj cp x 30], [Shares owned x 30], [MACD x 30], [RSI x 30], [CCI x 30], [ADX x 30]]

Portfolio Rebalancing

In portfolio balancing environment dimension of the state space vector is 240x1. The contents of the vector are as shown below.

[Adjusted Close/Open x 30], [High/Open x 30] , [Low/Open x 30], [Volume x 30], [MACD x 30], [RSI x 30], [CCI x 30] , [ADX x 30]

Table 1 contains description each of RL Environment State is below:

State Components	Description
Portfolio Balance	Available balance/funds in the portfolio account at current time step.
Adjcp	Adjusted close price of each of the Dow or ETF stock.
Shares Owned	Shares owned of each Dow stock.
Volume	Number of shares traded on any given day
MACD	**Moving Average Convergence Divergence (MACD)** - It is stock momentum indicator that identifies moving averages.
RSI	**Relative Strength Index (RSI)** - RSI quantifies the extent of recent price changes.
CCI	**Commodity Channel Index (CCI)** CCI compares current price to average price over a time window to indicate a buying or selling action.
ADX	**Average Directional Index (ADX)** ADX identifies trend strength by quantifying the amount of price movement.

Table 1: Components of States of Environment

In this reinforcement system, the agent is allowed to act on the portfolio account once every day. Once an action is completed by the environment, the date advances to next calendar date to fetch the state of market and combines it with state of portfolio account to advance to next state space.

4.2.4 RL Environment Component - Reward Function

Single and Multiple Stock Trading

Reward is the incentive mechanism for an agent to learn to improve action. The policy network which represents the intelligent agent needs to be trained to produce actions which can maximize the rewards. For every state of the environment, the possible action space is sufficiently large. If a matrix consisting of state space, action space and reward is constructed, the best action for a given state is the action with highest reward. An example matrix with action vector having 2 values is as below.

State	Action	Reward
S1	[0,0]	1
S1	[0,1]	3
S1	[1,0]	2
S1	[1,1]	1

Table 2: Rewards for State-Action pairs

Clearly in the above matrix, action [0,1] is the best action since it produces the highest reward. But the time and compute resources required to try every possible action in larger environments is impractical. Instead of trying every possible action, a global function approximator like neural network is used. To train this neural network there are multiple algorithms like A2C, PPO, DDPG etc. which interacts with the environment, collect the rewards back from the environment and train the neural network which is the policy network to produce actions which can get best rewards from the environment. While the training algorithms of policy network takes care of choosing the best action, the reward calculation is with the environment.

This paper defines the reward function as the change of the portfolio value when "action" is taken at "state" and arriving at "new state". The goal is to design a reward function that maximizes the total portfolio value. The reward is a number which indicates the total profit or loss that was achieved out of the action that was applied to the environment. This reward increases when actions result in decisions which leads to increasing the total asset value of the account. To direct the training of policy network in right direction, the environment needs to penalize the rewards for bad actions.

Example:

When an action to buy is suggested but there are no funds available in the account and an action to sell a stock which is not held in the account needs to be penalized.

Portfolio Rebalancing

The reward function for this environment was designed to maximize mean logarithmic cumulative returns. The reward function is defined as follows,

$$R_T = 1/T \sum_{t=1}^{T} log(1 + r_t)$$

Where

R_T is reward at time T

r_t is portfolio return at time t.

Portfolio return is calculated as follows,

$$r_t = \prod_{i=0}^{m} (Adjcp_{t,i}/Adjcp_{t-1,i} - 1)(w_{t,i})$$

Where

r_t is returns at time T,

m is number of stocks in the portfolio

w_t is weight vector at time t

Adjcp is adjusted closed price of the stock

Transaction cost at time T is calculated as follows, Transaction cost = $P_T \; * \; r_t$ * Total cost percentage

Where,

P_T Portfolio value at time T

4.3 Reinforcement Learning (RL) Policy Agent

An intelligent agent is the one which automates the portfolio management, and it replaces the human fund manager. This agent is nothing but a neural network. This intelligent agent also called as policy network will have to learn to take right kind of decisions for given market conditions. As mentioned in the previous section, there are multiple algorithms which can help train the policy network. Stable baselines library is a mature open-source library which implements these algorithms. This library is used successfully to train the policy network.

RL algorithms (Fig 5) are broadly divided based on the concept whether the agent has access to a model of the environment. By a model of the environment, it is meant a function which predicts state transitions and rewards. This paper focused on using different agent policy functions to create comparative results for analyzing which produced better total returns. The agent policies were selected on whether they work well for the financial stock trading action space (buy, sell, hold). Model-Free algorithms involve training function approximators like neural networks and they are suitable for environments with large state spaces like stock trading environments.

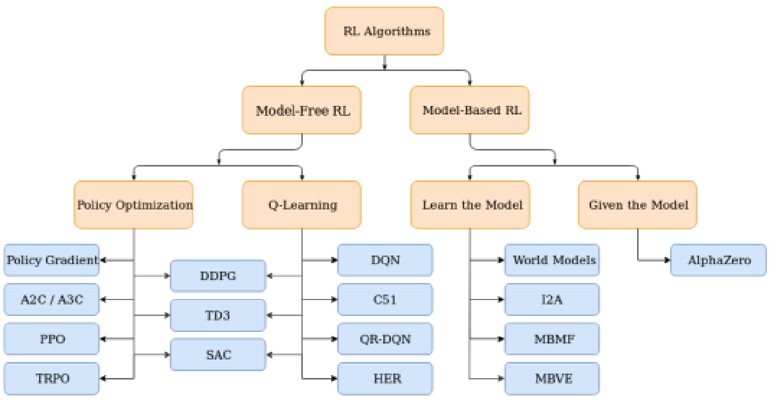


Fig 5: Taxonomy of RL Algorithms (not exhaustive)
Source: AI4Finance-LLC

There are many other available policy algorithms. The intent of this paper was not to conduct an exhaustive search but establish an ensemble framework that will allow for flexibility as market environment evolves.

The selection of algorithms used for this paper's study is based upon the related works referenced in Related Works papers [2][7][22]. Table 3 is an overview of the algorithms and for this paper only actor-critic algorithms were included in research. A2C, PPO and DDPG results are shown in this paper.

Algorithms	Input	Output	Type	State-action pair	Finance Use Cases	Advantages
A2C	State action pair	Q-Value	Actor-Crtic	Discrete and Continuous	All use cases	Stable, faster, better with large batch
DDPG	State action pair	Q-Value	Actor-Crtic	Continuous	Multiple Stock trading, Portfolio rebalancing	Better for high dimensional continuous action spaces
PPO	State action pair	Q-Value	Actor-Crtic	Discrete and Continuous	All use cases	Stable and Simple
SAC	State action pair	Q-Value	Actor-Crtic	Continuous	Multiple Stock trading, Portfolio rebalancing	Stable
TD3	State action pair	Q-Value	Actor-Crtic	Continuous	Multiple Stock trading, Portfolio rebalancing	Improved DDPG via twin Q Functions

Table 3: Researched Algorithms
(Note: Only A2C, DDPG and PPO results are included in paper)

4.3.1 Policy Agent Algorithm - PPO

In many settings where RL algorithm is used, the training data is generated on the fly while the agent explores the environment but in this portfolio management case, the market conditions are not affected by any action taken on the portfolio account. The training data from market is independent of the actions. So, the worry about exploring those regions in the environment which can result in learning a bad policy which is not useful is eliminated. But still there a worry about market data varying significantly. It can contain multiple distributions which are not similar to each other. As the policy network is trained on the market data, the learnt behavior of the network can be destroyed when it gets trained on a distribution which is not typical of the market behavior. Proximal Policy Optimization (PPO) addresses this issue by incrementally updating the policy network with an advantage function at every time step. In the mathematical formulation, $\pi\theta$ represents the policy network and A_t is an estimator of advantage function at time step t.

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$$

4.3.2 Policy Agent Algorithm - A2C

Advantage Actor-Critic (A2C) is a synchronous and deterministic version of A3C (Mnih et al., 2016) in the family of actor-critic methods. This is hybrid technique that combines value based (Q-Learning) and policy based (Policy Gradients) learning. It eliminates the limitation and takes the best from both methods. In this approach the Actor is a parametrized policy that defines how the best actions are selected and Critic on the other hand evaluates this action. The Actor then updates its policy based on Critic's evaluation. Advantage function in A2C is a low variance Q-value after taking State value off as equation shown below.

$$A\ (s, a) = Q\ (s, a) - V(s)$$

Fig 6 shows Asynchronous version (A3C) on the left and Synchronous version (A2C) on the right. In these methods multiple agents are trained with their own copies of environment, and each agent periodically updates global network. In asynchronous method different agents update global parameters at different times and may also have different versions of policies thereby aggregate updates might not be optimal. This is resolved by network shown on the right side of the diagram below in which it waits for each agent to finish an action on the environment before updating the global parameters. Next A2C restarts a new action on environment with all parallel agents having the same new state. Synchronous method proven to be performed better and it converges faster than asynchronous (A3C) method.

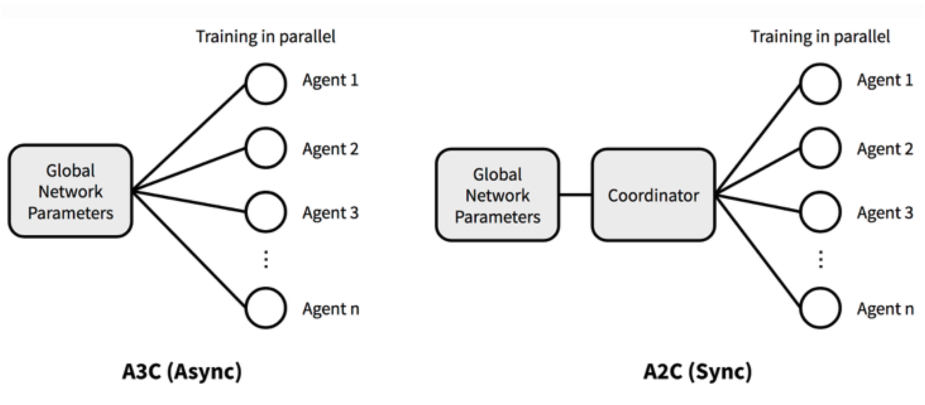


Fig 6: Architecture of A3C vs A2C
[Image source - Lil'Log's github website]

A2C can be used for Discrete or Continuous state actions. As with PPO, it is useful for either single stock trading or multi-stock trading scenarios. It is setup to work better with large batch sizes. This proves to be helpful when learning on stock market environment.

4.3.3 Policy Agent Algorithm - Deep Deterministic Policy Gradient (DDPG)

DDPG is a reinforcement learning technique that combines both Q-learning and policy gradients. Basic elements of DDPG are training actor-critic network models, target actor-critic network models and replay buffer. The actor is a policy network that takes the state as input and outputs the exact action, instead of a probability distribution over actions. The critic is a Q-value network that takes in state and action as input and outputs the Q-value. The replay buffer consists of the observation, action, reward and next observation [22]. Fig 7 is a process flow diagram that provides the DDPG approach for training.

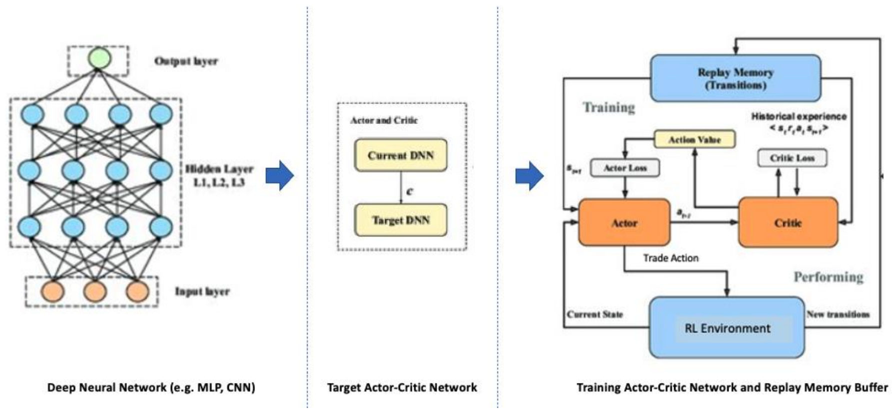


Fig 7: DDPG Algorithm Process Flow
[Source: spinningup.openai.com]

At each time step, the DDPG agent performs an action at state (s), receives a reward (r) at time (t) and transition to next state (st+1). The RL components (s, a, st+1, r) are stored in the replay buffer. A batch of N transitions are drawn from replay buffer and the Q-value yi is updated as [22]:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'}, \theta^{Q'})), i = 1, \cdots, N. \quad (1)$$

The critic network is then updated by minimizing the loss function L(θ Q) which is the expected difference between outputs of the target critic network Q0 and the critic network Q [22]:

$$L(\theta^Q) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1} \sim \text{buffer}}[(y_i - Q(s_t, a_t|\theta^Q))^2]. \tag{2}$$

DDPG can be used for continuous state actions. DDPG is an "off"-policy method. For multiple stock trades and portfolio rebalance, DDPG shows evidence of being useful for stock trading.

5 Results

The results from paper's research showed evidence that the "portfolio rebalance" modeling produced better accumulative returns and higher Sharpe Ratio comparative to the Dow baseline, Dow ETF single stock and Dow 30 multi-stock models. In Sections 5.2, 5.3 and 5.4 will show the results obtained for the respective stock trading scenarios (i.e., single, mult-stock and portfolio rebalance). In Section 5.1 the paper will outline a simple example how the RL stock trading model results demonstrate learning trading decisions.

5.1 Learnt pattern of Agent

One of the important aspect of results is to demonstrate the behavior of the agent in real world trading environment. To see how the intelligent agent performs at every trading cycle, the data about how many stocks were held in the account at every time step was collected and plotted. This is a particular case of AAPL stocks. The agent is demonstrating right buying and selling behavior at multiple places in this chart. Take for example at x-axis scale of around 2018-01. The agent bought around 500 stocks and sold it at around x-axis scale of 2018-04 when adjclose was higher thereby making a profit. This ideal behavior is not seen at every opportunity of low adjclose price. But few of this behavior is good enough to increase the asset value of the portfolio account.

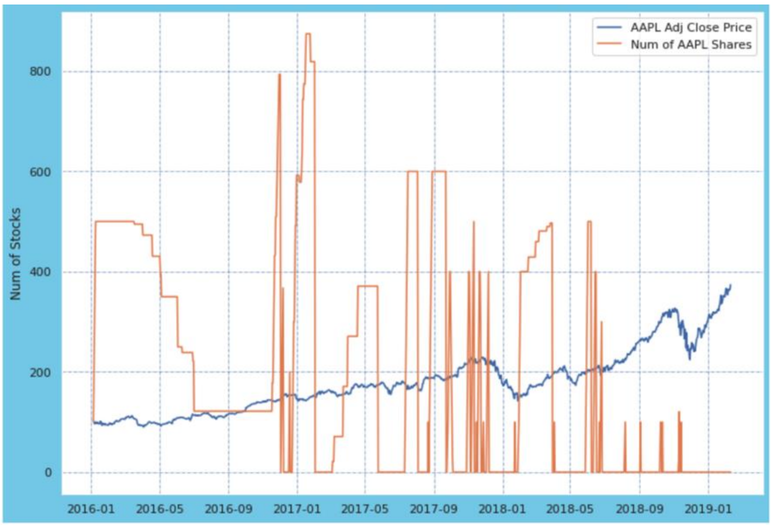


Fig 8: Trading decisions by RL agent

5.2 Dow ETF Single Stock Trading Results

As referenced in the Data Section, this paper conducted time-series analysis on the stock technical indicators to establish an initial RL state environment. Upon establishing initial RL environment state this paper ran RL policy models using the Dow Jones Index ETF (ticker symbol ^DJI) as an aggregate baseline to determine whether the time-series analytics showed evidence of being useful for more complex multiple stock trade and portfolio rebalancing.

Table 4 are sample output from the Dow ETF single stock trading modeling. As referenced in the Methods Section the goal for Dow ETF single stock trade modeling was to be an efficient approach of screening technical indicator features to be included in more complex Dow multi-stock and Dow portfolio rebalance modeling.

The cells highlighted in green show an example of evidence where the selected RL environment state action features and selected policy agent algorithm (PPO) may be useful for signaling stock trading decisions (i.e., buy, sell, hold) when compared to the ^DJI Baseline. The environment state space technical indicators that showed consistent evidence of producing higher Sharpe Ratios were primarily MACD, CCI and RSI.

Note: DDPG is not able to be used for single stock trade modeling and Ensemble was only utilized for multiple stock trading scenario that showed evidence of producing higher Sharpe Ratios.

Metric\Agent	Baseline	A2C	PPO
Annual Returns	13.45%	14%	15.14%
Cumulative Returns	101.24%	105.8%	117.02%
Annual Volatility	19.52%	19.2%	19.36%
Sharpe Ratio	0.74	0.78	0.82
Sortino Ratio	1.02	1.08	1.27
Stability	0.84	0.87	0.86
Max Markdown	-37.08%	-35.5%	-37.05%
Daily value at Risk	-2.40%	-2.40%	-2.37%
Alpha	0	0	0.01
Beta	1	0.98	0.98

Table 4: Backtest results from RL Models in Dow ETF single stock environment.

The results from the single stock trading modeling analysis were used to establish the environment state for the final multi-stock and portfolio rebalancing modeling. The process of leveraging single stock trading modeling is meant to be an iterative method for keeping the environment state relevant as the stochastic stock market evolves over time. Section 5.3 and Section 5.4 extend the modeling research using multiple stock trading environment, MLP and LSTM policy neural networks and hyperparameter tuning.

5.3 Dow 30 Multi-Stock Trading (Ensemble)

Multi-Stock portfolio account holds all 30 stocks of Dow Jones Index. The intelligent agent varies the proportion of stocks held in the account based on learnt behavior from past data of these stocks. The learning of intelligent agent is done by 3 different algorithms (A2C, PPO and DDPG) and the one with best Sharpe ratio is chosen for trading cycles. After 3 months of exposure to trading, the policy network is retrained for all the data until that point before employing it in trading again. The Ensemble agent for multiple stock trading environment was trained on data containing Dow Jones 30 stocks from Jan-2009 to Dec-2015 initially and then retrained every three months starting from day 1 up to 90 trading days that are already processed in the trading cycle. The initial investment amount was set to $100,000.

As shown in Table 5, with ensemble modeling, the portfolio account was able to get annual returns of 10.9%. As the research was conducted there was evidence that Portfolio Rebalance scenario outperformed the multi-stock trading approach. The rest of this paper is focused on Portfolio Rebalance scenario as outlined in the next section.

Metric\Agent	Baseline	Ensemble
Annual Returns	12.21%	10.9%
Cumulative Returns	78.24%	67.6%
Annual Volatility	20.24%	16.681%
Sharpe Ratio	0.67	0.72
Sortino Ratio	0.93	1.37
Stability	0.8	0.88
Max Markdown	-37.08%	-29.899%
Daily value at Risk	-2.48%	-1.172%
Alpha	0	0.12
Beta	1	0.05

Table 5: Back test results from RL Models in Dow ETF multi-stock environment.

5.4 Dow 30 Portfolio Rebalancing

To implement this form of multiple stock (portfolio) trading separate environment was created with the portfolio containing Dow 30 stocks. The purpose of this setting is to rebalance portfolio (change proportions of the stock) periodically in order to maximize returns. For this paper, rebalancing was done daily.

Three RL agents A2C, PPO and DDPG were trained and evaluated separately along with Ensemble which combines performance of these three agents and results are presented in the table 6. All models were trained on same data containing Dow series of 30 stocks from Jan-2009 to Dec-2015 initially. The agents trained for this duration are used for trading for 90 days and then they were retrained with the data from the original start date to already traded days and same process followed until end of the time series.

The different experiments that were carried out in the process of training it has been observed that on re-training these agents periodically they learn new patterns in the data, update their experience and hence their actions improve which in improves the cumulative returns in the portfolio. All agents were trained up to 80000 timesteps with an initial investment amount of $100,000. These agents were trained using OpenAI gym (stable baselines) library after tuning learning rate (0.00005) and entropy coefficient (0.0005) by leaving other hyperparameter configuration to default. The stable-baselines code base from OpenAI gym provides several policies to train RL agents. For the results presented in this paper agents have been trained using policies listed in Table 6 below:

Agent	Policy	Description
A2C	MlpLnLstmPolicy	This is actor-critic policy using normalized LSTM layer with MLP feature extraction.
PPO	MlpPolicy	This is actor-critic policy using MLP feature extraction.
DDPG	LnMlpPolicy	This is actor-critic policy using MLP with normalized layers.

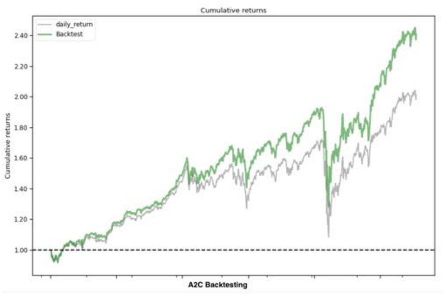
Table 6: Agent’s policy networks.

Table 7 shows backtesting result of the RL strategy learned by these agents for the trading period Jan 2016 up to July 2021 which is total 65 months of trading. As shown in the table Ensemble which combines performance of three agents performs better (Sharpe ratio 0.97) than other agents and it is a winning model of this experiment.

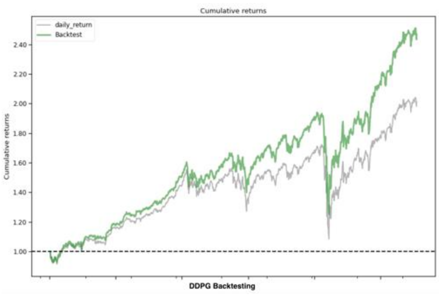
Metric\Agent	Baseline	A2C	DDPG	PPO	Ensemble
Annual Returns	13.45%	17.17%	17.73%	16.58%	18.40%
Cumulative Returns	101.24%	140.63%	146.98%	134.01%	154.92%
Annual Volatility	19.52%	19.19%	19.11%	19.11%	19.25%
Sharpe Ratio	0.74	0.92	0.95	0.90	0.97
Sortino Ratio	1.02	1.30	1.34	1.27	1.38
Stability	0.84	0.92	0.92	0.91	0.93
Max Markdown	-37.08%	-34.56%	-35.60%	-34.75%	-33.94%
Daily value at Risk	-2.40%	-2.34	-2.33%	-2.34%	-2.35%
Alpha	0	0.04	0.04	0.03	0.05
Beta	1	0.97	0.97	0.97	0.98

Table 7: Backtest results from RL Models in Portfolio rebalancing environment.

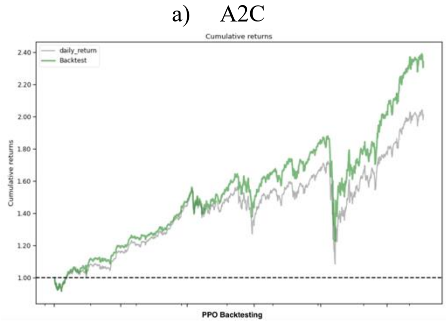
Fig-9 shows Backtesting plots obtained using python package pyfolio by Quantopian. On X-axis it is day of trading and on Y-axis is cumulative returns on the trading day.



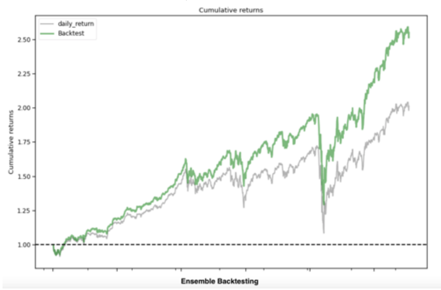
a) A2C



b) DDPG



c) PPO



d) Ensemble

Fig 9: Backtesting on RL strategies

As displayed in above figures RL strategy learned by these agents outperforms baseline which indicates that rebalancing portfolio periodically can help keep portfolio in good standing compared to doing nothing. The performance of all agents is close in terms of Sharpe ratio. This experiment found Ensemble gives better cumulative returns (154.92% in this case) compared to individual RL agents.

Ensemble Result in Detail

The ensemble method was employed for portfolio rebalancing trains all three models simultaneously, their performance was evaluated and the one with best Sharpe ratio is selected for next 90 days of trading. Table 8 shows details of experiment carried out for this paper. It depicts training, validation, and trading periods. Performance of agents in terms of Sharpe ratio for the given training period and selected RL agent based on Sharpe ratio. Last column trading Sharpe ratio shows the how selected agent performed on unseen data. It has a separate row for periods for which all agents were retrained after every 90 days of trading.

Retraining Periods	Training Period		Validation Period		Trading Period		Sharpe Ratio of trained agents and selected agent				Sharpe
Series Period	From	To	From	To	From	To	A2C	PPO	DDPG	Selected	Trading Sharpe Ratio
1	2009-01-02	2013-11-21	2013-11-22	2015-12-30	2016-01-04	2016-05-10	0.89	0.9	0.92	DDPG	0.95
2	2009-01-02	2014-02-25	2014-02-26	2016-05-10	2016-05-11	2016-09-16	0.82	0.74	0.75	A2C	1.08
3	2009-01-02	2014-05-27	2014-05-28	2016-09-16	2016-09-19	2017-01-26	0.73	0.78	0.74	PPO	1.38
4	2009-01-02	2014-08-25	2014-08-26	2017-01-26	2017-01-27	2017-06-06	0.82	0.74	0.81	A2C	1.56
5	2009-01-02	2014-11-21	2014-11-24	2017-06-06	2017-06-07	2017-10-12	0.83	0.84	0.83	PPO	1.84
6	2009-01-02	2015-02-25	2015-02-26	2017-10-12	2017-10-13	2018-02-22	1.05	0.94	0.98	A2C	1.83
7	2009-01-02	2015-05-27	2015-05-28	2018-02-22	2018-02-23	2018-07-02	1.19	1.2	1.11	PPO	1.46
8	2009-01-02	2015-08-25	2015-08-26	2018-07-02	2018-07-03	2018-11-07	1.41	1.43	1.46	DDPG	1.56
9	2009-01-02	2015-11-23	2015-11-24	2018-11-07	2018-11-08	2019-03-21	1.31	1.29	1.29	A2C	1.29
10	2009-01-02	2016-02-25	2016-02-26	2019-03-21	2019-03-22	2019-07-30	1.45	1.44	1.34	A2C	1.33
11	2009-01-02	2016-05-25	2016-05-26	2019-07-30	2019-07-31	2019-12-05	1.41	1.31	1.34	A2C	1.29
12	2009-01-02	2016-08-24	2016-08-25	2019-12-05	2019-12-06	2020-04-16	1.15	1.26	1.13	PPO	0.71
13	2009-01-02	2016-11-22	2016-11-23	2020-04-16	2020-04-17	2020-08-24	0.58	0.56	0.45	A2C	0.79
14	2009-01-02	2017-02-24	2017-02-27	2020-08-24	2020-08-25	2020-12-31	0.59	0.57	0.6	DDPG	0.85
15	2009-01-02	2017-05-25	2017-05-26	2020-12-31	2021-01-04	2021-05-12	0.72	0.64	0.71	A2C	0.97
16	2009-01-02	2017-08-24	2017-08-25	2021-05-12	2021-05-13	2021-07-21	0.79	0.82	0.84	DDPG	0.97

Table 8: Ensemble training, validation, and trading

6 Discussion

6.1 Challenges and Recommendations

The study proved interesting in understanding how Reinforcement Learning using Markov Decision Process (MDP) can be potentially useful for stochastic environment modeling. The growing availability of RL packages (e.g., OpenAI and Stable Baselines) allows for more focus on RL research vs. creating algorithms from scratch.

The paper also proved helpful in understanding that the RL modeling approach was not about predicting the stock market prices but rather the use of rewards on "buy", "sell" and "hold" decisions when training to produce useful model for making trading decision eventually. In addition, this paper's observation that using feature engineering methods such as time-series analysis for establishing environment state space shows evidence of being beneficial in obtaining models with higher Sharpe Ratio's. The amount of time for hyperparameter tunning the neural networks and exhaustive research of potential technical indicators far exceeded available time for this paper's research.

It is the paper's consensus that further analysis would be prove useful and may lead to building confidence in RL model's abilities and apply them to actual stock trading. This includes further research on available financial stock trading indicators and gaining more in-depth financial market domain knowledge. The limitations are understanding that this modeling approach would require significant "paper testing" prior to deploying in real world trading. In addition, the model provides thousands of trade decisions to be completed over the investment period. This would require deploying this model in conjunction with an automated brokerage trade algorithm.

This modeling does not include data about stock appreciation for each individual stock with respect to the buying price of the stock. At every instance of trade if this data is made available to the agent, it may provide better decision generation. This needs to be explored as part of future research.

6.2 Ethical Consideration

Stock trading is an extremely difficult, complex, and tedious job for any portfolio manager. The movement of stock price depends on several factors like investors' sentiments, fundamentals of companies, trading volumes, market news, politics, policy decisions, etc. The job of portfolio managers is to keep a close watch on what is happening in the market, investments, politics etc., estimating the price movements in the market so that they can make accurate decisions for their clients to earn better returns.

The market generates huge volumes of data and at remarkably high velocity due to volumes of transactions that occur every minute so analyzing this information in a timely manner to take right decisions at the right time is very much critical. The automated agents can perform such analysis more efficiently than humans. Therefore, portfolio managers have started preferring automated agents to take full advantage of the market dynamics.

The goal of automated agent using reinforcement learning presented in this paper is to train machines to do such analysis and take appropriate actions in order to assess risks and maximize returns. They can be fully automated to perform analysis and to take automatic actions such as buy, sell or hold based on analysis. The ethical issue is that these ML (Machine Learning) models, specifically RL agents, evolve with the data and can take biased/illegal decisions if not evaluated periodically or someone can just deploy to manipulate market.

These models can execute trading transactions in large volumes in a short span of time and can potentially disrupt the market if bad decisions are taken by the agent either accidently or intentionally. Stock markets are regulated markets, and such actions are heavily penalized once detected. It is obvious that accidental behavior of these agents can be managed by following risk management principles and by following the best software practices during the development phase and by monitoring live behavior of the agents.

6.3 Future works

There are significant areas available to extend the initial research and the research conducted in related works. Future works in consideration include the use of sentiment-related indicators and social media analytics to improve upon RL modeling and at minimum, may provide useful in assessing market volatility. Potential of reframing the stock market analysis using CNN to capture daily market activity as images and use for training the policy agent neural networks.

7 Conclusion

This paper concludes that Markov Decision Process (MDP) and Reinforcement Learning show evidence of producing applicable stock trading decision models. The study's models show, via Sharpe Ratio, evidence of outperforming the baseline Dow Jones Index. This paper researched policy agent algorithms such as A2C, PPO and DDPG, and each appears to perform comparatively better to each others at various stock trading periods. This reflects in the results when using an ensemble of all the policy agent algorithms at three-month model training intervals. Using Portfolio Rebalance scenario, this paper's results show that the "ensemble method" produced the highest Sharpe Ratio and highest cumulative portfolio final balance. The feature engineering using time-series also contributed to the selection of environment state space that aided in producing this paper's results.

Given the evolving stochastic nature of the stock market, the requirement for future efforts is that continual feature engineering and an ensemble method be applied when performing future training of RL stock market trading models.

Acknowledgments. Authors are extremely thankful to all who helped make this project possible and especially thankful to Dr. Santeree (Advisor), David Stroud (Co-Advisor) and Dr. Jacquelyn Cheun (Capstone Professor) for their support and guidance throughut on challenging capstone project like this.

References

1. Chen, L., & Gao, Q. (Oct 2019). Application of deep reinforcement learning on automated stock trading. Paper presented at the 29-33. doi:10.1109/ICSESS47205.2019.9040728.
2. Dang, Q. (2019). Reinforcement learning in stock trading. Advanced computational methods for knowledge engineering (pp. 311-322). Cham: Springer International Publishing. doi:10.1007/978-3-030-38364-0_28.
3. Gyeeun Jeong and Ha Kim. (2018), "Improving financial trading decisions using deep q-learning: predicting the number of shares, action strategies, and transfer learning," Expert Systems with Applications, vol. 117.
4. John Moody and Matthew Saffell. (2001). Learning to trade via direct reinforcement, IEEE Transactions on Neural Networks, vol. 12, pp. 875–89.
5. Chong, T., Tai-Leung, N., Liew, W., & Khim-Sen, V. (2014). Revisiting the performance of MACD and RSI oscillators Journal of Risk and Financial Management, vol. 7, pp. 1–12, 03.
6. Jae Won Lee, Jonghun Park, Jangmin O, Jongwoo Lee, & Euyseok Hong. (2007). A multiagent approach to Q-learning for daily stock trading. IEEE Transactions on Systems, Man and Cybernetics. PartA,Systems and Humans,37(6),864 877. doi: 10.1109 /TSMCA. 2007.904825
7. Hieu, L. T. (2020). Deep reinforcement learning for stock portfolio optimization. doi:10.7763/IJMO.2020.V10.761
8. Zhu, H., Wang, Y., Wang, K., & Chen, Y. (2011). Particle swarm optimization (PSO) for the constrained portfolio optimization problem. Expert Systems with Applications, 38(8), 10161-10169. doi:10.1016/j.eswa.2011.02.075
9. Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., & Fujita, H. (2020). Adaptive stock trading strategies with deep reinforcement learning methods. Information Sciences, 538, 142-158. doi:10.1016/j.ins.2020.05.066
10. Wang, W., Li, W., Zhang, N., & Liu, K. (2020a). Portfolio formation with preselection using deep learning from long-term financial data. Expert Systems with Applications, 143, 113042. doi:10.1016/j.eswa.2019.113042
11. Yaman, I., & Erbay Dalkılıç, T. (2021). A hybrid approach to cardinality constraint portfolio selection problem based on nonlinear neural network and genetic algorithm. Expert Systems with Applications, 169 doi:10.1016/j.eswa.2020.114517
12. Kimoto, T., Asakawa, K., Yoda, M., & Takeoka, M. (1990). Stock market prediction system with modular neural networks doi:10.1109/IJCNN.1990.137535
13. Carta, S., Corriga, A., Ferreira, A., Podda, A. S., & Recupero, D. R. (2020). A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. Applied Intelligence (Dordrecht, Netherlands), 51(2), 889. doi:10.1007/s10489-020-01839-5
14. Ciaburro, G. (2018). Keras reinforcement learning projects : 9 projects exploring popular reinforcement learning techniques to build self-learning agents / giuseppe ciaburro (1st ed.). Birmingham, UK: Packt Publishing Ltd.
15. Kimoto, T., Asakawa, K., Yoda, M., & Takeoka, M. (1990). Stock market prediction system with modular neural networks doi:10.1109/IJCNN.1990.137535
16. Soleymani, F., & Paquet, E. (2020). Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath. Expert Systems with Applications, 156, 113456. doi:10.1016/j.eswa.2020.113456
17. Aboussalah, A. M., & Lee, C. (2020). Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization. Expert Systems with Applications, 140, 112891. doi:https://doi-org.proxy.libraries.smu.edu/10.1016/j.eswa.2019.112891

18. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., . . . Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning.
19. Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., . . . Chintala, S. (2020). PyTorch distributed: Experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment, 13*(12), 3005-3018. doi:10.14778/3415478.3415530
20. W. Muller and H. Schumann. Visualization methods for time-dependant ¨ data-an overview. In Proceedings of the 2003 Winter Simulation Conference, pages 737–745, 2003.
21. S. F. Silva and T. Catarci. Visualization of Linear Time-Oriented Data: A Survey. In Proceedings of the International Conference on Web Information Systems Engineering (WISE), pages 310–319. IEEE, 2000.
22. Hongyang Yang , Xiao-Yang Liu, Shan Zhong and Anwar Walid. Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy