EEE3096 Individual Project

Data-Driven Computing: Application of Reinforcement learning in Algorithmic Trading

Ayobola Abiola
Student No. 1906429683
H652 BEng Hons Electronics & Computer Engineering
School of Engineering

Supervisor: Prof Alex Yakovlev

Table Of Contents

Abstract

Data-Driven Computing is a field in computational analysis that is relatively Novice/Novel , It works by using data obtained over a period of time to derive predictive outcomes. Data-driven computing can often work in tangent with machine learning in order to make industrial decisions based on the predictive outcome of the Data Driven algorithm. In today's financial market, the majority of deals are now entirely automated, and algorithmic stock trading has become a standard. Many difficult games, such as Chess and Shogi, Reinforcement Learning (RL) agents have shown to be formidable opponents. The historical price series and movements of the stock market may be viewed as a complicated imperfect information environment in which we aim to optimise return - profit while minimising risk. In this report I will study different forms of Machine learning, focusing on, and building a Reinforcement Learning algorithm for trading that will specifically use the GME financial database, talking about the relevance of Data quality and quantity and its effect.

1. Introduction

We are approaching an era where the importance of data is skyrocketing, Data is being generated in higher volumes, complexity, and variety. Data affects every aspect of modern civilization, from business to healthcare. Data-Driven Computing utilises machine learning as well as the huge volume of data constantly being created to implement systems for different purposes and analyses [1].

1.1 Machine learning

There are three main forms of machine learning algorithms, Supervised learning, Unsupervised Learning and Reinforcement Learning.

- Supervised learning (SL): One of the most fundamental forms of machine learning is supervised learning which is trained on labelled data. Even though precise labelling of data is required for this approach to operate, supervised learning may be incredibly effective when utilised under the right situations.

- Unsupervised Learning (UL): Unsupervised learning, on the other hand, lacks labels to deal with, leading to the Unsupervised learning algorithm to identify patterns in the data that were previously unknown, for example identifying homogeneous segments on an image. Unsupervised learning algorithms do not know the absolute truth but the ability to deal with unlabelled data is a benefit.

- Reinforcement learning: Reinforcement learning is a framework that learns to make decisions under uncertainty to maximise a long-term benefit through trial and error. These decisions are made sequentially, and earlier decisions affect the situations and benefits that will be encountered later. Since there are no absolute truths, the model learns through feedback of its own experience. The tasks that Reinforcement Learning tries to accomplish are not only of a different nature but tend to be more complex than those addressed by SL and UL.

1.2 Data

Data is a key component in all forms of machine learning, and without it machine learning is not possible. Based on the data, machine learning will grow its experience and ability to make decisions [2], therefore the properties of the data will have a significant role in the success of the algorithm. Some properties to consider would be:
Volume of the data - Vast amounts of data are created every millisecond as the world's population grows and technology becomes more accessible, but too much data can lead to incredibly long training times as well as require a lot of manpower. However, if the data is limited this can cause inaccuracy in the algorithm and may require data augmentation (modifying/distorting the input, mostly randomly, so that the diversity in the training data increases [3])
Data veracity: This relates to confidence and accuracy in the data being analysed. Data quality is determined by a number of factors, including where the data was obtained, how it was acquired, and how it will be processed.
Authenticity: The authenticity of a user's data determines how trustworthy and significant the information is. Low-veracity data typically comprises a significant percentage of non-valuable, 'noisy,' and nonsensical data, which will hinder an organization's analysis. High veracity data, on the other hand, has a large number of records that are useful to an organization's analysis and contribute meaningfully to the final outcomes.[4]

1.3 Motivation

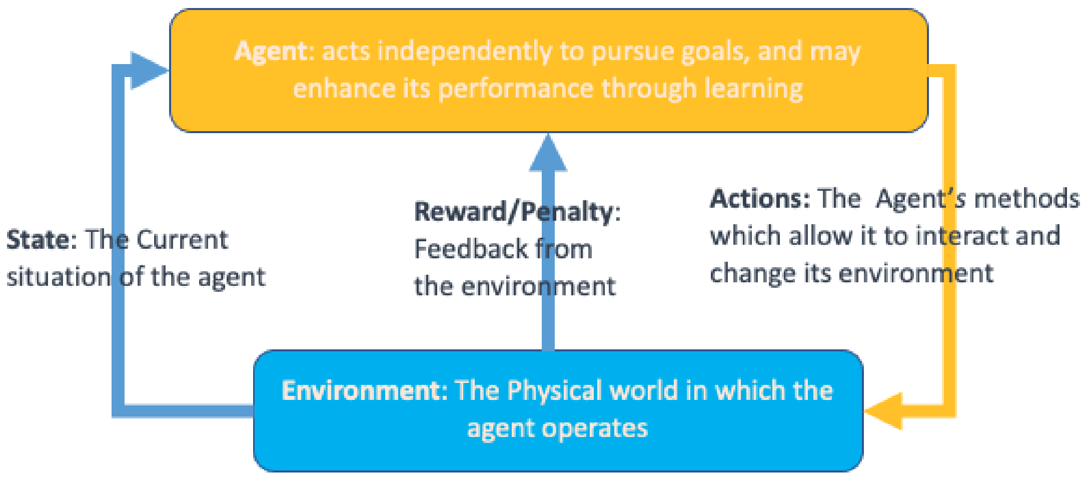
Stock trading is the act of purchasing and selling a company's stock on a financial market. The idea is to maximise the capital's return on investment by using recurring buy and sell orders to take advantage of market volatility. When one buys at a cheaper price than one later sells, profit is made. The rising complexity and dynamical characteristic of stock markets are two of the most well-known issues in the finance business. Expertly devised trading methods frequently fail to generate positive returns in all market situations.[5]
Investment firms and hedge funds require a profitable automated stock trading approach. It is used to optimise capital allocation and investment performance, such as predicted return on investment. Estimates of possible return and risk can be used to maximise return, designing a lucrative strategy in a complicated and dynamic stock market is difficult. The clear question that I will try to answer is: "Can we make a successful trading agent That uses the Data-driven Reinforcement learning and Datasets from previous stock trends and patterns?

2. Theoretical Background

2.1 Reinforcement Learning (RL)

Reinforcement Learning is a field of artificial intelligence used for creating self-learning autonomous agents. attempts to impersonate the way human beings learn new things through interaction with an environment [19]. For example, to teach a child how to catch a ball, it will learn through feedback from parents or friends. Reinforcement learning has the goal of creating an agent that can make decisions in a complex/uncertain environment whilst maximising the long-term benefit. Since Reinforcement learning agents learn through interacting with their environment its applications are rather broad and adaptable with potential to revolutionise global industries.

Figure 1: Reinforcement learning process diagram


Agent: acts independently to pursue goals, and may enhance its performance through learning
State: The Current situation of the agent
Reward/Penalty: Feedback from the environment
Actions: The Agent's methods which allow it to interact and change its environment
Environment: The Physical world in which the agent operates

In Reinforcement Learning every time the agent performs an action it receives either a reward or penalty. The agent monitors the condition of the environment in an iteration of a Reinforcement Learning problem and takes action depending on its observations. Therefore, as shown in Figure 1, the agent receives a reward/penalty and the environment transitions into a new state. [3]. The long-term implications of the agent's activities are important as it must think about which actions t will be beneficial in the long term. Some behaviours may result in the agent receiving large rewards right away, only to be followed by very small rewards or even penalties. It's equally possible that the contrary is true.[3]

- If the Reinforcement Learning problem is an episodic task, where an episode is defined as the sequence of interactions from an initial state to a terminal state, the agent's purpose is to maximise the total predicted cumulative reward gathered across an episode.

- Meanwhile, a continuous task is a challenge that is specified across an indefinite horizon. Considering the total reward might reach to infinity, the agent will aim to maximise the average reward.

2.2 Markov Process and Markov Decision Process (MDP)

A key ingredient in Reinforcement Learning is the Markov Decision Process which is used to model the environments mathematically. Markov Decision Process model is the framework used to model sequential decision-making problems. If the probability of advancing to state St+1 depends only on the current state St and not on the prior state, $S_1, S_2, \cdots, S_{t-1}$, then the sequence of states is Markov. [3,20]

$$P[S_{t+1} | S_t] = P[S_{t+1} | S_1, S_2, \cdots, S_t].$$

The Markov Process also known as the Markov chain is a tuple (S,P), where

- S is a (finite) set of states
- P is a state transition probability matrix

$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s].$$

The Markov process follows the following dynamics: It begins at some state s_0 and transitions to a successor state s_1 deduced from P_{s0s1}. Then we go on to s_2, which is derived from P_{s1s2}, and so on.
This dynamic is represented as the following:

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$$

The Markov Decision Process is defined by ***(S, A, P, R, γ)***. It introduces actions (***A)*** to the Markov Process which gives the impression of having control over the Markov Process. [21]

- ***S*** is a (finite) set of states
- ***A*** is the set of actions
- ***P*** is the state-transition probability
- ***R*** is the reward
- ***γ*** is the discount factor

Unlike the other Markov processes, when it comes to the Markov Decision Process the next state and rewards are dependent on what action the agent takes.

$$\mathcal{P}^a_{ss'} = \mathcal{P}(s'|s,a) = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

$$\mathcal{R}^a_s = \mathbb{E}[\mathcal{R}_{t+1} | S_t = s, A_t = a]$$

2.3 Markov Decision Process to Address Reinforcement Learning

Rewards(***R***) are short term, so if the agent chooses an action that leads to a decently high reward, it may be missing out on an overall greater total reward in the long run. The long-term total reward is known as Return (***G)***

$$\mathcal{G}_t = \sum_{i=0}^{N} \gamma^k \mathcal{R}_{t+1+i}$$

The discount factor (***γ***)t ranges between 0 to 1 and its purpose is to reduce the weightage of future rewards as future benefits are uncertain. As important as it is to consider future rewards to enhance the Return, it is also necessary to restrict the contribution of future rewards to the Return (since you can't be 100 percent confident of the future).

A policy (***π***) is a distribution over actions given states that expresses the reasoning behind a decision (picking an action). It describes how an RL agent behaves. In formal terms, a policy is a probability distribution over a set of actions given the current state, i.e., it determines the likelihood of selecting an action at state [21].

The State-Value Function (V_π) is defined as the expected discounted return when starting in a specific state. However, the State Value Function in an MDP is defined for a policy.

$$\mathcal{V}_\pi(s) = \mathbb{E}_\pi[\mathcal{G}_t | \mathcal{S}_t = s]$$

A quantity used frequently in reinforcement learning is the Action Value Function. This quantity answers the question "What would the expected cumulative return be if action “a” is taken while in the current state and follows (***π***) thereafter for all states?"

$$\mathcal{Q}_\pi(s,a) = \mathbb{E}_\pi[\mathcal{G}_t | \mathcal{S}_t = s, \mathcal{A}_t = a]$$

2.4 Bellman Equation

Long-term reward in a particular activity is equal to the current reward plus the expected reward from future actions committed at a later period, according to the Bellman Equation. If the Bellman Equation is not used when the starting position is changed, it will be unable to retrace its steps.

$$V(s)=\max_a(R(s,a)+\gamma V(s'))$$

- **(s):** current state where the agent is in the environment [22]
- **(s'):** the next state the agent reaches after taking action(a) at state(s) [22]
- **(V)**: Numeric value of a state which helps the agent to find its path. [22]
- **(R):** Reward which the agent gets after performing an action(a). [22]
- **R(s):** reward for being in the state s [22]
- **R(s,a):** reward for being in the state and performing an action a [22]
- **R(s,a,s'):** reward for being in a state s, taking an action a and ending up in s' [22]

γ: the discount factor determines how much the agent prioritises rewards in the distant future relative to those in the immediate future. It has a value **between 0 and 1**. **Lower value** encourages **short–term** rewards while **higher value** promises **long-term reward**

2.5 Neural Networks/Deep Learning

When the state space or action space are too big to be entirely known, neural networks are useful function approximators. A standard neural network incorporates four main components:

- Inputs
- Weights
- Bias or Threshold
- Output

To estimate a value function or a policy function, a neural network can be employed. To put it another way, neural networks can learn to map states to values or state-action pairs to Q values. We can train a neural network on samples from the state or action space to learn to predict how valuable those are relative to our target in reinforcement learning. An alternative would be to use a lookup table to store, index, and update all possible states and their values, however this is impossible with very large problems [24].

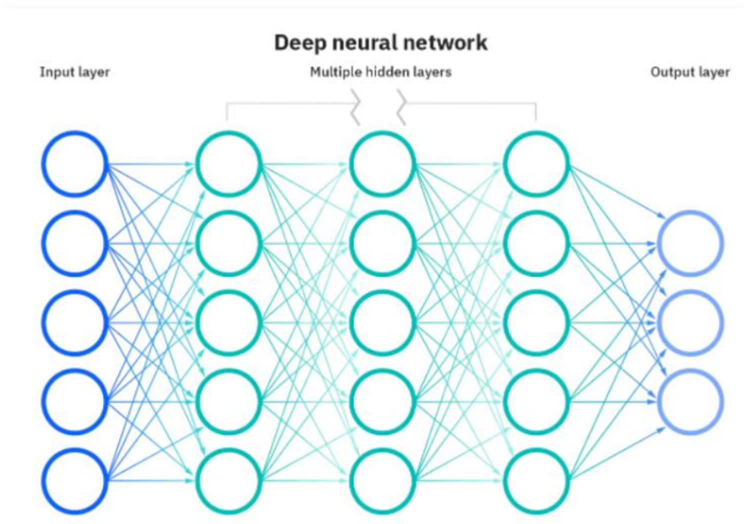


Figure 2: IBM Representation of a deep learning algorithm

Deep learning algorithms are neural networks with more than three layers, which include the inputs and outputs [25]. These algorithms are known to be highly scalable because they are equipped to handle large data sets and automates features that would normally require human intervention. generally, deep learning is used in more complicated applications. In addition, deep Learning can take advantage of labelled data but is also capable of performing when the data is not labelled by observing patterns in the data; a deep learning model can cluster inputs appropriately.

2.6 Deep Reinforcement Learning

Deep Reinforcement Learning, an amalgamation of Deep Learning and Reinforcement learning, uses a neural network as a function approximator for the Q value on large data sets allowing for complicated tasks to be solved. [23]. Reinforcement learning could be applied to huge datasets by using a neural network as a function approximator [23]. Deep reinforcement learning is an amalgamation of Deep Learning and Reinforcement learning techniques which allow for complicated tasks to be solved. When applying deep reinforcement learning in the financial sector one of these 3 learning approaches are to be considered:

- Critic-only Deep Reinforcement Learning
- Actor-critic Deep Reinforcement Learning
- Actor-only Deep Reinforcement learning

3 Literature review

Throughout this section, Reinforcement learning, and trading literature sources will be analysed. The key focus areas to be observed are data quality and availability, the partial observability of the environment, and the exploration/exploitation dilemma [6]. This review will look at Critic-only Deep Reinforcement Learning, Actor-only Deep Reinforcement Learning and Actor-critic Deep Reinforcement Learning.

The book 'Mastering Reinforcement learning with python [3]' gives an idea of the fundamental challenges with using reinforcement learning in finance by outlining the difficulty of the task. It mentions that whilst it may take a human one to two hours of play time to become relatively decent at a game a Reinforcement learning agent would require millions if not billions of game frames to reach a similar level. Furthermore, the difficulty of games pale in comparison to becoming a successful stock trader which requires years of financial experience, possibly a Ph.D. and even with those success is not guaranteed. Other challenges that were highlighted by this book include [3]:

- Financial market efficiency: "Financial markets are highly efficient, and it is challenging, if not practically impossible to predict the market. In other words, the signal is frail in the financial data and is almost always riddled with noise".[3]
- Financial markets are dynamic: "A trading strategy that is profitable today might be short lived as competitors will affect the effective of the strategy".[3]
- Data size: "The real-world data is not that big to easily detect the low signal in it." [3]
- Financial markets are real: "As the financial markets are real-world processes, simulations need to be created in order to collect huge amounts of data needed for training the agent, especially due to the noisiness of the environment. Not only are simulations going to be imperfect they can also be costly resulting in a smaller net profit" [3]

3.1 Critic-only Deep Reinforcement Learning (DQN)

The Critic-only method is the most widely published method in this field of study when compared to Actor-only and Actor-critic Deep Reinforcement Learning. The Critic-only method aims to learn the best action-selection policy that maximises projected future reward given the current state using a Q-value function. However, one significant drawback is that it can only function with discrete and finite state and action spaces, which is inconvenient for a big portfolio of stocks because prices are, of course, continuous. If applied to multiple stocks and assets, there will be limitations due to the state and action growth being exponential. Moreover, the definition of reward function in this context is crucial; critic-only approaches being sensible to the reward signals it receives from the environment. [5]

The paper 'Application of Deep Reinforcement Learning on Automated Stock Trading [8]' uses the same logic behind the success of Deep Q Network concerning learning to play Atari like games to develop a trading agent. The trading environment is treated like a game where the reward is attempted to be maximised signalled by profit [10]. The author employs a recurrent neural network as the foundation of DRQN, which can interpret temporal sequences, and a second target network to help stabilise the process is proposed. The dataset utilised to train and test the agent is the S&P500 ETF 19 year daily closing prices provided by Yahoo Finance. The first 5 years were utilised for training and the remainder for testing the agent. As a result, a state representation is defined by a series of adjusted close price data across a 20-day sliding frame. [8,10]. A buy and hold strategy, used by human long-term investors, and a DQN trader with random activity are the baseline benchmarks for / against the agent. Figure 3 shows the best Agent is successful in beating the baseline models. The base system annual predicted return is approximately 22-23 percent, which although adequate pales in comparison to the 60% returns of other examined agents in this research. This indicates that algorithmic trading is a viable option [11].

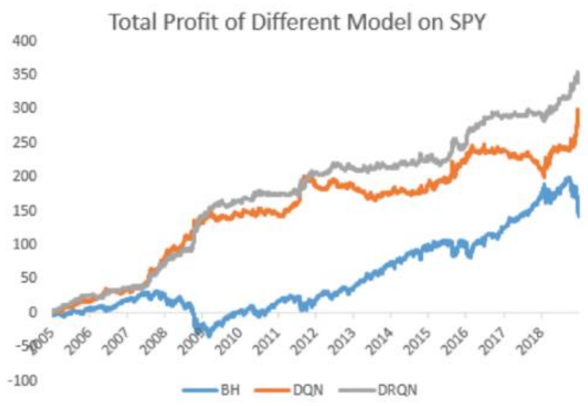


Figure 3: “Profit Curves of BH, DQN and DRQN on the SPY Test Dataset” [12]

However, there are glaring limitations on the study:

- A singular stock is used as a benchmark, causing questions to arise if the approach is viable when applied to larger portfolios; some stocks can become successful whilst others could lose all the allocated budget.
- The reward function is relatively weak; it does not address risk like other evaluated papers therefore it may be unpredictable to run this model on real data. [10]
- The agent's capabilities are severely limited with the ability to purchase only one stock at a time; additional actions might be introduced to enable for the purchase or sale of more shares. The feature representation is substandard; raw price data contains a lot of noise, which may have been mitigated by using technical analysis features. [10,11]
- Regarding the data the smaller size of the training data compared with the testing data causes concerns with appropriate representation. The dataset is also between 2006-2018, during which the market crash of 2008 occurred, so the agent will presumably lack knowledge on how to tackle such outlying data.

This paper is a solid foundation that shows possibility when it comes to utilising deep reinforcement learning in trading. However, comparisons with other trading agents would be desirable.

In ‘Deep Reinforcement learning for Trading [7]’ The experiment is similar but replaces stocks with contracts. In this the author demonstrates that baseline models are outperformed by the DQN system Algorithms.

3.2 Actor-only Deep Reinforcement learning

In Actor-only Deep Reinforcement learning methods the action space may be generalised to be continuous since a policy is directly learned - this is the fundamental advantage of this approach [10]. The lengthy training time required to acquire optimum policies is a problem encountered with our actor-only strategy. This occurs because, in trading, learning requires a large samples to be successful. Otherwise, individual poor actions will be considered acceptable if the overall rewards are high. Unlike the critic only approach there is no numerical value given to specific states and actions that can tell us differently through an expected/predictive outcome. Although in the case of the survey ‘Reinforcement learning in financial markets-a survey [13]’ a faster convergence of the learning process is observed.

In the paper ‘Deep Direct Reinforcement Learning for Financial Signal Representation and Trading [14]’, the question of beating experienced traders is attempted to be answered, Written in 2017 the author claims this was the first instance of using deep learning for ‘real-time’ financial trading. In the experiment they restructure a standard Recurrent Deep Neural Network to allow for simultaneous environment sensing and recurrent decision making in an online setting. Recurrent Neural Network for the RL component is employed, synonymous to the previous study examined; however, it uses Deep Learning for feature extraction, together with fuzzy learning techniques [15,16,17], to reduce the uncertainty of the input data. Instead of using technical indicators, the suggested strategy operates the aforementioned system for raw data processing and feature learning to generate intelligent estimates using the RL system in order to optimise profit. Back propagation is also introduced to handle vanishing gradient problems in deep architectures. To further clarify, it uses extra links from the output to each hidden layer during the backpropagation making more room for the gradient to flow [10]. Although the paper

introduces novel ideas to solve its proposed question, this does not imply that the performance using only raw data and clever data processing would beat current state-of-the-art models that make use of technical analysis [10].
One important flaw in this study is the inconsistency with which it answers the fundamental topic that it raises and addresses. The experiments lack comparison to human trading professionals' performance, and there is no discussion of this in the conclusion.

The paper 'Quantitative Trading on Stock Market Based on Deep Reinforcement Learning' discusses a deep actor-only approach to quantitative trading. Its key contributions include a thorough examination of the benefits of using a deep neural network (LSTM) over a fully connected one, as well as the influence of various technical indicator combinations on performance on daily-data Chinese markets. This study demonstrates that a more in-depth strategy is preferable, and that selecting the appropriate technical indicators makes a difference when it comes to profit. [10,15,18]

3.3 Actor-critic Deep Reinforcement Learning

The third type of RL framework, the actor-critic framework, seeks to train two models at the same time: the actor, who learns how to get the agent to respond in a certain state, and the critic, who assesses how well the selected action was. State-of-the-art actor-critic DRL algorithms like PPO (in an actor-critic context like it was presented in the original work) or A2C addressing complicated settings proved to be one of the most effective in the literature.
Despite its supposed effectiveness the method has stayed among the least studied methods in DRL for trading with little exploration. There are a few things that Actor-critic Deep Reinforcement Learning does better than previously mentioned RL types:

- It addresses well-known problems of applying RL to complex environments.

- Over time, the actor learns to take better actions and the critic gets better at evaluating those actions. The actor-critic approach has proven to be able to learn and adapt to large and complex environments, and has been used to play popular video games, such as Doom. Thus, the actor-critic approach fits well in trading with a large stock portfolio.

3.4 Thoughts

In the financial world, Deep Reinforcement Learning is becoming increasingly popular since machine learning algorithms can considerably advance financial investment strategic data. The machine's processing speed can greatly enhance strategy adaptability and the capacity to derive market features from real-time trading signals. .
The Critic-only method has been researched substantially more than the others in relation to application to Finance and Trading despite its drawback of only working with finite, discrete state, and action spaces. This may be because in all the reviewed papers, the daily time frame chosen to trade on, led to an unexpected smaller s data set size. This means the Critic-Only method was still usable without showing its weaknesses. Daily time frame is what was used due to access, as although trading occurs by the second/minute without paying for access to this data it would be difficult to acquire.

4. Methodology

I introduce setups in this part, which include state, action spaces, and reward functions. Advantage Actor-Critic (A2C) techniques and Proximal Policy Optimization are two Reinforcement Learning algorithms used in this project. The goal is to build a custom trading agent using reinforcement learning, and specifically the stable baseline package, which offers another method to begin learning about and implementing reinforcement learning. Afterwards, the custom bot is trained with data from the GME Trading stocks loaded into the trading environment in hopes that the agent can learn how to trade the stock and return a profit on top of the portfolio. The comparison will be between two different reinforcement learning algorithms belonging to the actor-critic Reinforcement Learning framework, A2C and PPO. I aim to collect results on how they both perform under the same conditions as well as show how data impacts each algorithm.

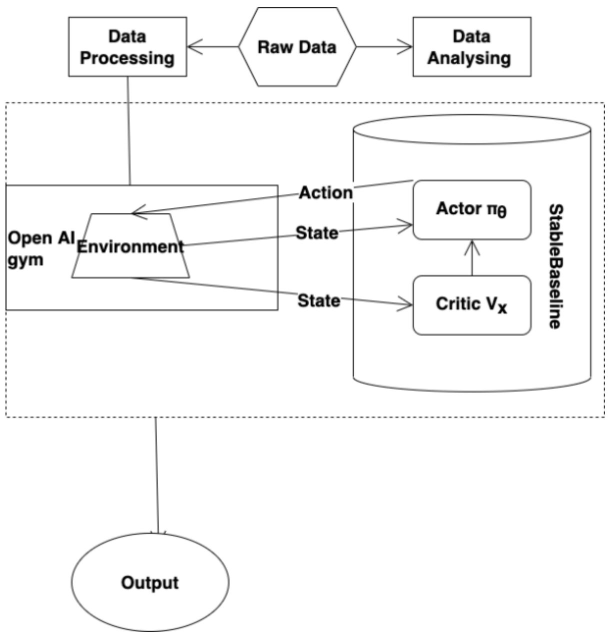


Figure 4: Our deployment's architecture for Experiment (Actor Critic model)

4.1 Actor Critic Model Overview

The actor critic algorithm is made up of two networks (the actor and the critic) that collaborate to solve a problem. The Advantage Function calculates the agent's TD Error or Prediction Error at a high level. At each time step, the actor network selects an action, while the critic network assesses the quality or Q-value of a particular input state. The actor uses this information to instruct the agent to seek out good states and avoid bad states as advised by the critic network.

4.2 Reinforcement Learning Algorithm

The Actor Critic algorithm used to train the agent works by updating the policy in real time [27]. It comprises two models: one is an actor network that produces policy, and the other is a critic network that assesses the effectiveness of the chosen action in the current condition. Subsequently, the actor uses this information to instruct the agent to seek out good states and avoid bad states as the critic network learns whether states are better or worse by computing the temporal difference (TD) error. The estimated value of all future rewards from the current state S is the TD Target. The Critic Network calculates the value of the following state S' using the function V(s).

4.3 A2C (Advantage Actor Critic)

In A2C (Advantage Actor Critic) the TD_Error is usually equal to the Advantage, thus is depicted as the Prediction error of our Agent. The Advantage function shows if a state is better or worse than expected. If a state exceeds expectations, the advantage is greater than 0. Meanwhile if the advantage is less than 0, the state is worse than expected. During implementation, the agent explores its environment whilst the critic network attempts to drive the advantage function towards 0. The critic will most certainly make huge errors at the start of the learning process, causing the calculated TD error to be completely inaccurate. This is due to the critic having no knowledge of the environment in the beginning. As the critic makes more and more accurate predictions, the calculated TD error (Advantage) grows increasingly precise allowing the actor to use it to determine if a move was good or poor.

4.4 PPO (Proximal Policy Optimization)

Whilst PPO is also an Actor Critic Algorithm, it focuses on ensuring that a new update to the policy does not cause drastic differences when compared to the prior policy. Thus, one can expect a smoother training process along with removing the possibility of an illogical course taken by the agent that is unrecoverable. PPO takes short samples of interactions with the environment and reuses the same batch, discarding the previous experiences and using the new updated policy as the basis for the next policy [28]. PPO comes in two principal varieties: PPO-Penalty and PPO clip. This essay/experiment concentrates on PPO-Clip - the most common PPO version in OpenAI [28].

4.5 Data Stitching

The data stitching procedure was straightforward. Initially, the GME CSV file was inputted into the Jupyter Notebook as a **Pandas** Dataframe [28]. Considering the data source was simple, eliminating null values and obsolete fields like "Volume Currency" and "Weighted pricing." was the only cleaning required. To establish the Reinforce Learning environment and input our data, I used the **OpenAI** gym package. In this setting, the data frame was separated into serial order and then constructed an algorithm (agent) to execute the trade [28] . For the training and validation purposes, the dataset was divided into an 80/20 split. The entire reward and profit earned by the agent would be the algorithm's output. I presented the trading history after the training and visualised the agent's action (long/short) in relation to the price [28].

4.6 Coding Steps

Importing dependencies

Firstly, install and import dependencies which in this case would be the gym/gym any trading stable-baselines, NumPy, pandas and matplotlib.

- **Gym/gym_anytrading**: a set of Open AI Gym environments for trading algorithms that use reinforcement learning. They can be found applying a variety of actions such as Buy, Sell, Hold, Enter, Exit, and so on. For the purpose of this experiment, I will be using only buy and sell.

- **Stable-baselines** provides a set of default policies that can be used with most action spaces [26]. Success in reinforcement learning will be dependent on this as it permits access to algorithms such as PPO and A2C.

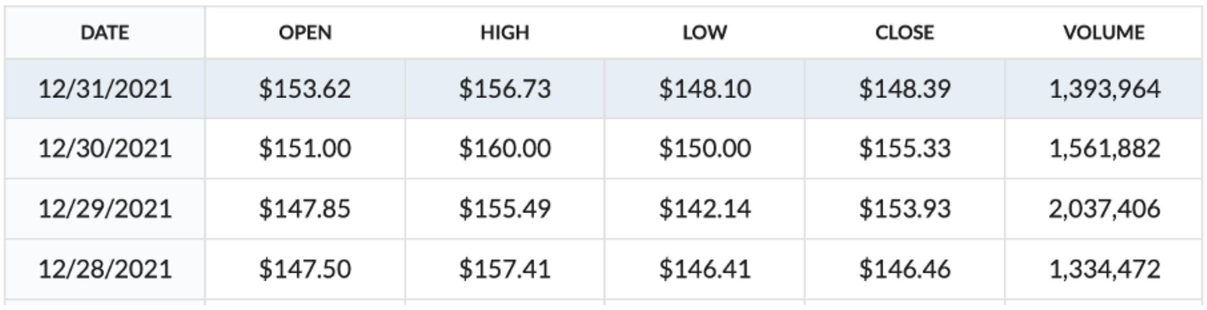
- **NumPy, pandas and matplotlib**: These are packages used to assist with data analysis. For example, they make loading data from external sources like text files and databases much easier.

Bringing in GME Data using Pandas

Secondly, track and select the data to train the agent and test the performance. The GME financial dataset is the daily historical stock price of GameStop, in this project I chose the dataset to be from the range of January 2008 to December 2021. I will be using 80% of this data as the training set which allows me to use the last 20% for testing/validation. The data set contains:

- Date: The days of trading
- Open: The opening price of the stock
- High: The highest price of the day
- Low: The Lowest price of the day
- Close: The closing price of the day
- Volume: The number of stocks traded during that day

As seen in figure 5



DATE	OPEN	HIGH	LOW	CLOSE	VOLUME
12/31/2021	$153.62	$156.73	$148.10	$148.39	1,393,964
12/30/2021	$151.00	$160.00	$150.00	$155.33	1,561,882
12/29/2021	$147.85	$155.49	$142.14	$153.93	2,037,406
12/28/2021	$147.50	$157.41	$146.41	$146.46	1,334,472

Figure 5: The fixed columns in the GME financial Dataset

The trading agent will look at the open and close indicators primarily to attempt to locate a pattern as each day passes for example spotting if the stock is losing value or if the stock is gaining value by the day, then using the recognised pattern to inform its decision during the testing process.
But the data selected does not have to be limited to daily historical data. If one could acquire data that is recorded in minutes or seconds, the outcome would be a more sophisticated model. Although, it is ideal that the model mimics the frequency that one chooses to trade at.

To import the data to the code, we use the standard pandas read csv functionality specifying the path to the dataset. Needless to say, the data does not have to be GME but could be any choice of historical stock or crypto data such as Bitcoin.

The next step when setting up for trading is to convert the date column in our dataset into a date time because it is an object which would not allow it to work with the gym any trading environment. To do this we use the pandas .**to_datetime** which function effectively takes our existing date column and passes it through the datetime column and then overwrites the existing date column.
However, to pass the date through, the column needs to be set as the index for the **gym any trading** environment to receive the data.

Building Environment

Now it is time to concentrate on the trading environment where the agent learns to trade which the data passes through; **env.sigal_features** can be applied to look at the various signal features that it contains to be used for trading. In this instance, it has the price and the change in price from the previous time step. To build the environment, one key line is written **env = gym.make**:. This combined with importing **gym_anytrading** enables a template environment to be set up by using **stocks-v0** to implement a stock environment. Next, parameter to be passed through is the stock data that has been pre-processed, df=df along with two other customizable keyword arguments.

- Frame_bound – shows the portion of data included within the environment.

- Window_size – the number of previous price and price sets available to the trading agent.

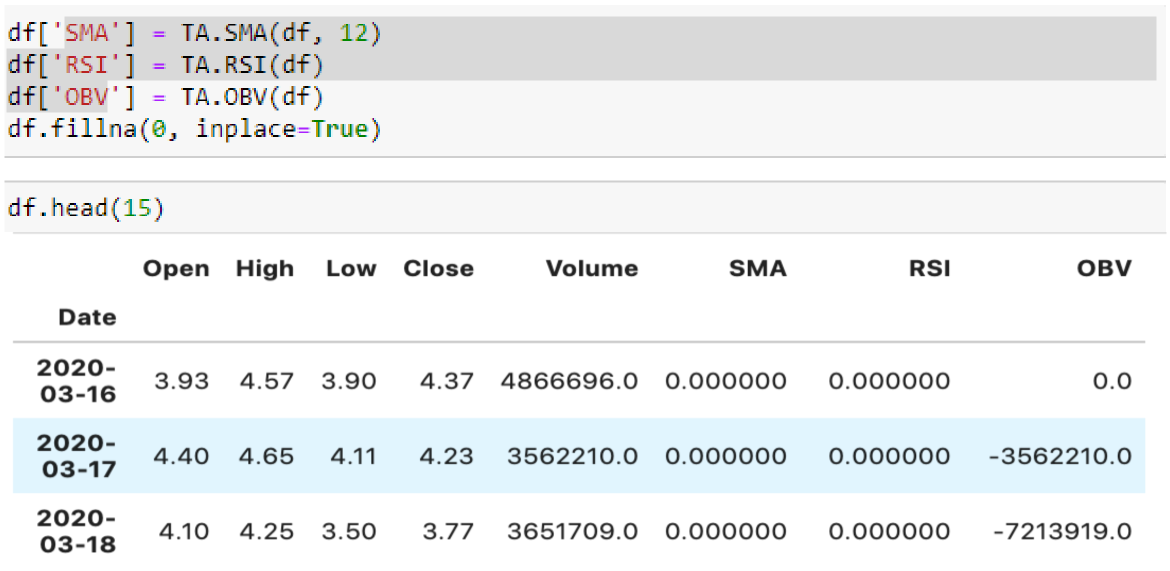
Testing the trading Environment

State = env.reset – sets the state back to its original hence getting back our signal features. With most open ai gym environments using **env.action_space** which hold two actions that can be taken which in this case is either buy or sell.

Training the training Agent

Many alternative reinforcement learning algorithms are included in the stable-baslines library. The algorithm used for Reinforcement Learning is easily exchangeable as it depends on which is imported total timesteps is the total number of steps the agent will take in any given environment. Total timesteps can span multiple episodes, indicating that there is no upper limit to this value.

5. Results



```
df['SMA'] = TA.SMA(df, 12)
df['RSI'] = TA.RSI(df)
df['OBV'] = TA.OBV(df)
df.fillna(0, inplace=True)
```

```
df.head(15)
```

	Open	High	Low	Close	Volume	SMA	RSI	OBV
Date								
2020-03-16	3.93	4.57	3.90	4.37	4866696.0	0.000000	0.000000	0.0
2020-03-17	4.40	4.65	4.11	4.23	3562210.0	0.000000	0.000000	-3562210.0
2020-03-18	4.10	4.25	3.50	3.77	3651709.0	0.000000	0.000000	-7213919.0

Figure 6: adding customised environments with technical indicators

Unfortunately, the initial try did not go as planned, and the agent lost all its money as seen in Figure 7. To remedy this, I added customised technical indicators [SMA, RSI and OBV] as seen in Figure 6. The purpose of technical indicators is to extract the most minor correlated indicators from a data batch then go through normalisation to be fed into the Reinforcement Learning agent to possibly increase performance.

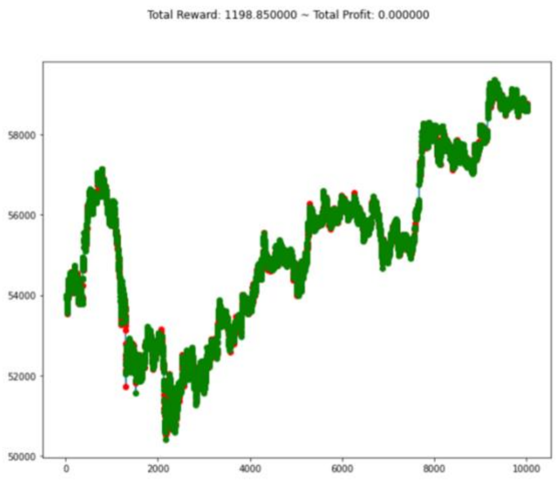
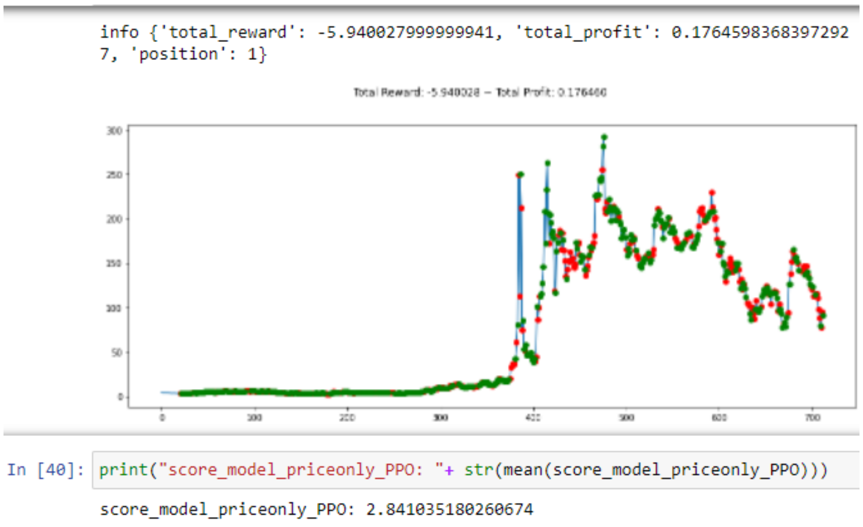


Figure 7: The initial testing of my trading agent

- **RSI:** The relative strength index measures the magnitude of changes in price to help find out which stock is overbought or oversold. The RSI works on value system of 0 to 100. If the stock from my GME dataset is rated as 70 one day, then I can say that the stock is being overvalued where as if GME data set is 30 and under then the stock is undervalued or oversold

- **SMA:** The simple moving average is used to smooth the data in the GME dataset once implemented into an array to help eliminate noise whilst identifying trends

- **OBV:** The on-balance volume uses the volume column in the GME dataset to make predictions depending on the volume changes



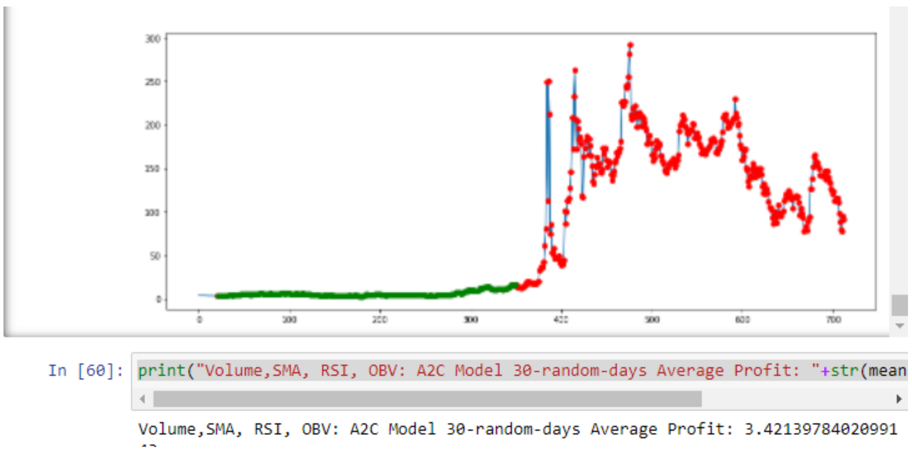
```
info {'total_reward': -5.940027999999941, 'total_profit': 0.1764598368397292
7, 'position': 1}
```

```
In [40]: print("score_model_priceonly_PPO: "+ str(mean(score_model_priceonly_PPO)))

score_model_priceonly_PPO: 2.841035180260674
```

Figure 8: Score model price only PPO trading graph

Once the technical indicators were added the average profit made by the Score model price only PPO as seen in Figure 8 was 2.841. Although the results were varied, meaning sometimes high profits are gained but other times the profit is low, there was a 2.481x profit.

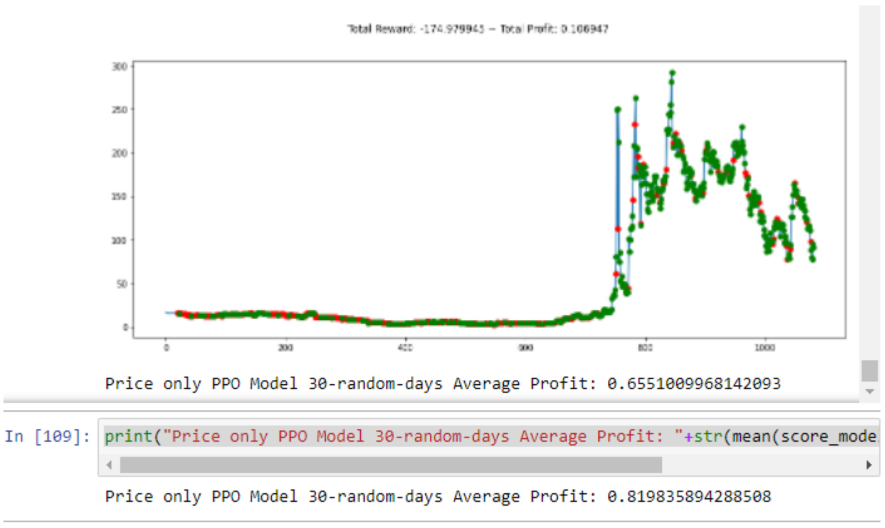


```
In [60]: print("Volume,SMA, RSI, OBV: A2C Model 30-random-days Average Profit: "+str(mean

Volume,SMA, RSI, OBV: A2C Model 30-random-days Average Profit: 3.42139784020991
```

Figure 9: Score model no price A2C trading graph

With respect to the Score model no price A2C the average profit was even greater as evidenced in Figure 9 Whilst this seems impressive, upon further comparison of the trading graph in Figure 9 with Figure 8 the agent appears to only be performing the action linked to the red dot after a certain period. This may have coincidently led to a greater result.



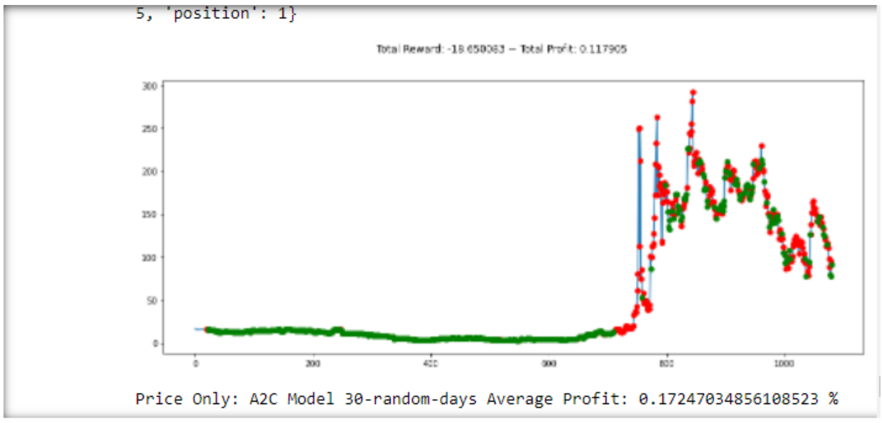
```
Price only PPO Model 30-random-days Average Profit: 0.6551009968142093
```

```
In [109]: print("Price only PPO Model 30-random-days Average Profit: "+str(mean(score_mode

          Price only PPO Model 30-random-days Average Profit: 0.819835894288508
```

Figure 10: Score model Price Only time series PPO trading graph

The timeseries PPO model performed adversely from its non-time series counterpart - the model lost money. As shown in Figure 10 the model's profit is 0.8198x meaning that approximately twenty percent of the starting allowance was lost during the trading period.



```
5, 'position': 1}

Price Only: A2C Model 30-random-days Average Profit: 0.17247034856108523 %
```

Figure 11: Score model price only time series A2C trading graph

The time series A2C model performed dissimilar from its non-time series counterpart; the model also lost money. Figure 11 reports the model's profit is 0.1725x, so the model lost over eighty percent of its starting allowance. Secondly the model in Figure 11 performed both actions of longing and shorting unlike in figure 9.

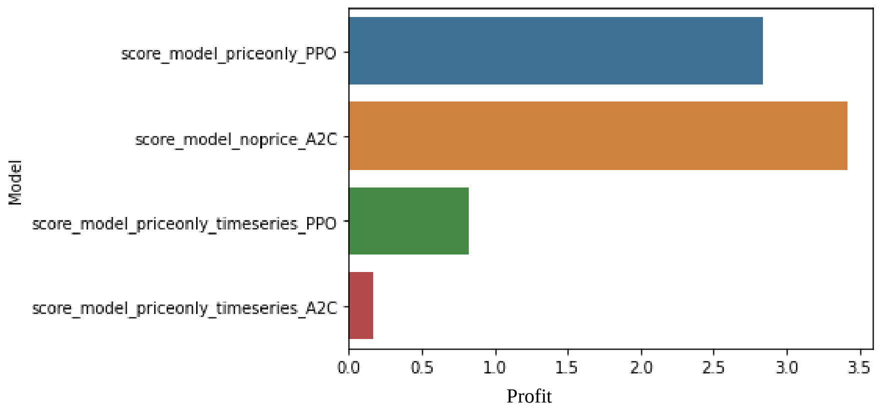


Figure 12: The four trading Models Results placed into a bar graph

5.1 Discussion/Conclusion

In the trading history, the red dot means the “long” action was taken while green means “short”. As identified in Figure 12 the highest performing model was the score model no price A2C with the score model price only PPO coming in second, This result differed from my expectation that the PPO model would outperform the A2C because earlier research revealed that PPO performed better than A2C in other environments such as the lunar landing and the cartpole. A potential explanation is that PPO focuses on assuring that a new update to the policy does not cause drastic differences when compared to the prior policy. Therefore, the policy often grows less random over time as the update rule encourages it to take advantage of incentives it has already discovered since it is a very reward driven algorithm. Yet, if you review Figure 9, after a certain point the agent just continuously longs. Even though it reaches the highest profit, its method of achieving this would not be applicable in all trading environments. I am unable to ascertain the cause of this discrepancy, but it may be due to the model requiring longer training time or a larger dataset. Further research will need to be conducted to understand why this event is occurring. The results in Figure 8 and Figure 9 were a lot larger than anticipated leading to the possibility of an error with the process used to test the agents.

Due to the drastically high profits of the **score_model_priceonly_PPO** and **score_model_noprice_A2C**, I created two more models that used time series (seen in Figure 9 and Figure 10). Time series analysis is used for non-stationary data—things that are constantly fluctuating over time or are affected by time -- hence I assumed it could be applied to the stock market. When the two time series models are compared, the PPO model easily outperforms the A2C model. Whilst the PPO model outperforming the A2C aligns with what I predicted the models both result in an overall loss; this means the agents were not successful However, the result in terms of profit/loss was more in line with what I expected this leads me to believe the agent may become successful if given a lot more training time and possibly fed a larger data set with smaller intervals.

During the process of this project, I was able to investigate the core concept behind Reinforcement Learning, Deep Reinforcement Learning and how both can be applied in trading environments implemented using python libraries such as stable baselines and open ai gym. I was able to collect data from the GME stock trading website and use a variety of techniques to clean and organise the financial data.

Although I was not successful in building a trading agent that performed solidly, my results and inferences lead me to believe the PPO agent is better than the A2C. This may have occurred due to the lack of training time therefore for a more successful trading agent, training should occur for a significant **time period** rather than the five hours applied in this experiment. I propose that this research be undertaken in the future on a cloud framework with scalable computing power and parallel computing capabilities. Although this architecture would surely be pricey, it would allow future researchers to work more efficiently and fulfil the project's goal.

6. What is Next?

Further areas of research include applying the Ensemble Deep Reinforcement Learning Trading Strategy. Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Deterministic Policy Gradient are three actor-critic based algorithms included in this strategy (DDPG). [23]

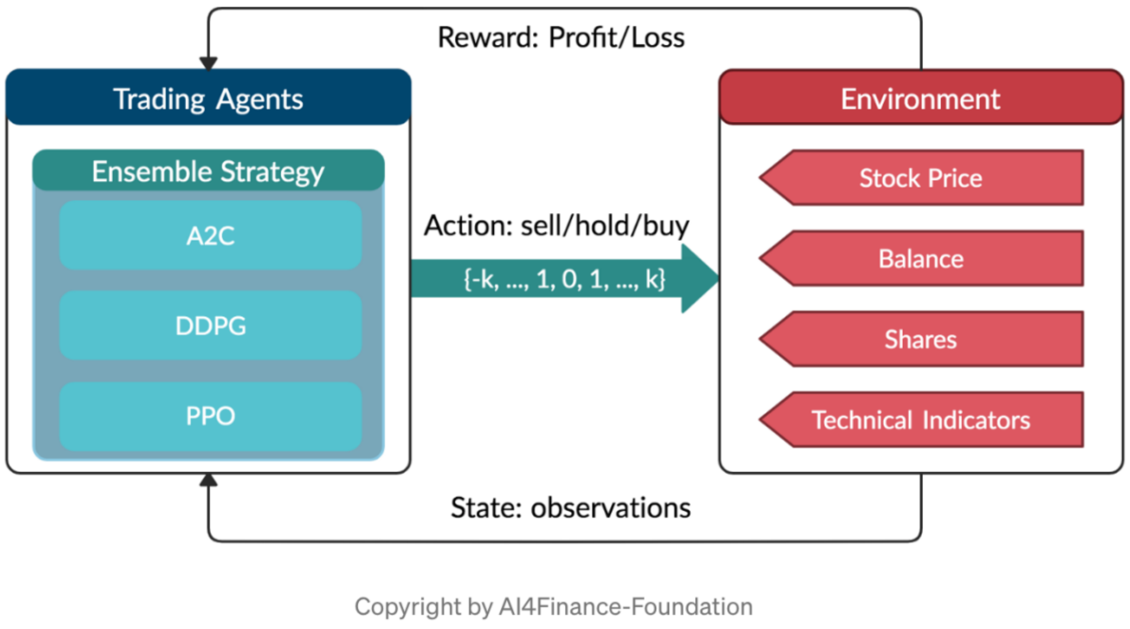


Figure 13: Ensemble Deep Reinforcement Learning Trading Strategy

It combines the greatest elements of the three algorithms, allowing it to adapt to changing market situations with ease. “The Sharpe ratio is used to compare the performance of the trading agent using different reinforcement learning algorithms to the Dow Jones Industrial Average index and the classic min-variance portfolio allocation approach”. [23]

7. References

[1] Advances in Intelligent Systems and Computing 1319 Jagdish Chand Bansal Lance C. C. Fung Milan Simic Ankush Ghosh Editors

[2] Understanding The Importance Of Data For Machine Learning by M Shehzen Sidiq

[3] mastering reinforcement learning with python by Enes Bilgin

[4] What is Data Veracity? Written by Indicative Team

[5] Xing Wu, Haolei Chen, Jianjia Wang, Luigi Troiano, Vincenzo Loia, and Hamido Fujita. Adaptive stock trading strategies with deep reinforcement learning methods. Information Sciences, 2020.

[6] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[7] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep reinforcement learning for trading. The Journal of Financial Data Science, 2(2):25–40, 2020.

[8] Zhuoran Xiong, Xiao-Yang Liu, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading. arXiv preprint arXiv:1811.07522, 2018.

[9] Lin Chen and Qiang Gao. Application of deep reinforcement learning on automated stock trading. In 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), pages 29–33. IEEE, 2019

[10] Deep Reinforcement Learning in Quantitative Algorithmic Trading: A Review Tidor-Vlad Pricope The University of Edinburgh Informatics Forum, Edinburgh, UK, EH8 9AB

[11] Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. arXiv preprint arXiv:1807.02787, 2018.

[12] Lin Chen and Qiang Gao. Application of deep reinforcement learning on automated stock trading. In 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), pages 29–33. IEEE, 2019.

[13]Thomas G Fischer. Reinforcement learning in financial markets-a survey. Technical report, FAU Discussion Papers in Economics, 2018.

[14]Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. IEEE transactions on neural networks and learning systems, 28(3):653–664, 2016.

[15]George J Klir. Fuzzy sets. Uncentainty and Information, 1988

[16]Nikhil R Pal and James C Bezdek. Measuring fuzzy uncertainty. IEEE Transactions on Fuzzy Systems, 2(2):107–118, 1994
[17]Chin-Teng Lin, C. S. George Lee, et al. Neural-network-based fuzzy logic control and decision system. IEEE Transactions on computers, 40(12):1320–1336, 1991.

[18]WU Jia, WANG Chen, Lidong Xiong, and SUN Hongyong. Quantitative trading on stock market based on deep reinforcement learning. In 2019 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2019.

[19] A Mathematical Introduction to Reinforcement Learning Xintian Han

[20] A Mathematical Introduction to Reinforcement Learning Xintian Han

[21] Understanding Markov Decision Process (MDP) Towards Training Better Reinforcement Learning Agents by Rohan Jagtap

[22] https://www.geeksforgeeks.org/bellman-equation/

[23] Deep Reinforcement Learning for Automated Stock Trading Using reinforcement learning to trade multiple stocks through Python and OpenAI Gym | Presented at ICAIF 2020 by Bruce Yang

[24] https://wiki.pathmind.com/deep-reinforcement-learning

[25] https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

[26]https://stable-baselines.readthedocs.io/en/master/modules/policies.html

[27] Deep Reinforcement Learning for Trading Zihao Zhang, Stefan Zohren, and Stephen Roberts Department of Engineering Science, Oxford-Man Institute of Quantitative Finance, University of Oxford

[28] Exploring Reinforcement Learning and BitCoin Trading- Dennis Lee