– Independent Work Report Spring 2020 –

Reinforcement Learning for Stock Trading

Victor Hua
Adviser: Karthik Narasimhan

Abstract

Machine learning methods for trading in financial markets have been explored in recent years and are of interest to researchers and financial analysts alike. Reinforcement learning in particular shows promise in this field. This project investigates the application of reinforcement learning to equities trading in comparison to a baseline trading strategy known as LONG. The goal is to build intuition as to how RL should be applied to stock trading and to determine if RL can outperform the LONG strategy. Different observation spaces, reward functions, and trading intervals are evaluated, as well as the performance of three common reinforcement learning algorithms on five different stocks. We find that a DQN trained on 2016-2018 closing price data can marginally outperform the LONG strategy on average across all stocks.

1. Introduction

Reinforcement learning (RL) algorithms have been utilized to perform certain tasks and reach performance benchmarks with great success. A popular application of RL is to Atari games such as Space Invaders to achieve superhuman results as well as to classic turn-based games like chess; most notably, AlphaZero by DeepMind has managed to decisively beat existing chess engines as well as chess grandmasters using reinforcement learning [2]. Financial markets are beginning to see a shift from human traders to algorithmic trading, with trading bots executing an increasing proportion of trades, yet existing work on using RL for trading is surprisingly limited and fairly new. The development of robust, consistent trading bots using RL can revolutionize the algorithmic trading world for both retail investors and institutional investors.

Previous work on RL for trading manage portfolios of several stocks, but the results of trading individual stocks has yet to be presented. Furthermore, the effects of different observation spaces, action spaces, and reward functions are not explained. Results averaged over several episodes are also

not presented. In this paper, we evaluate the performances of three common RL algorithms—Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C)—on equity against the baseline "LONG" strategy, which is equivalent to a measure of overall strength and direction of individual stocks.

We assess these algorithms on five blue-chip stocks that are traded with high frequency. The goal is to develop a trading agent that can consistently outperform LONG on individual stocks over several episodes and develop an intuition as to why certain observation spaces, action spaces, and reward functions work well while others don't.

2. Previous Work

Previous work has focused on maintaining portfolios of several stocks and evaluating their performance against the Dow Jones Industrial Average, a common stock market index, and min-variance portfolio allocation using Deep Deterministic Policy Gradient (DDPG) [9]. DDPG outperformed both the Dow Jones and min-variance allocation. RL has also been successfully utilized to identify trends in the stock market and optimal times to long or short [8]. Q-learning on high-frequency trading data has also been explored and shown to significantly outperform existing optimization strategies [6]. Most recently, the performance of several RL algorithms, namely DQN, Policy Gradients (PG), and A2C on various asset classes, including equities, were evaluated against time series momentum strategies using metrics such as Sharpe ratio, expected return, and positive trade return percentage [10]. DQN was found to perform the best across all asset classes except equities, which was optimized by a "Long-only" strategy where the only trade executed is a maximally long position at the beginning of the trade window.

Clearly, previous works have implemented differing strategies on varying data across different asset classes. Rewards, observation spaces, and action spaces are also defined very differently. This paper steps away from portfolio management and evaluates the performance of RL algorithms on individual stocks as well as the way different rewards, observation spaces, and action spaces affect RL algorithm performance in equity markets.

3. Approach

3.1. Markov Decision Process

Stock trading can be formalized as a Markov Decision Process (MDP). An MDP consists of an agent, a state space S, an action space A, and reward function $R(s,a)$. At each time step t, the agent's environment is represented as a state $s_t \in S$, and the agent selects an action $a_t \in A$. The agent is rewarded $R_t = R(s_t, a_t)$, and the process proceeds to the next time step $t+1$. Thus, the reward the agent receives for doing a_t at time t is dependent only on the state s_t and not any of the states $s_0, \ldots, s_{t-1}$. The agent seeks to maximize its total reward R_T, defined to be the sum over all of its incremental rewards at each time step throughout the process. For the purposes of stock trading, the action space can be a simple variant of $\{\text{Buy}, \text{Sell}\}$, but formulating the reward function and state space in a way that optimizes the agent's performance will prove to be difficult.

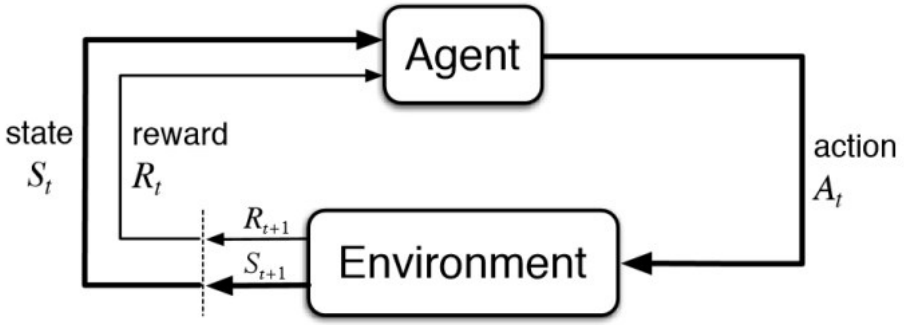

Figure 1: Formulation of a time step [1].

3.2. Baseline Strategy

Typically a trading strategy's performance is compared to the market's performance. There are several metrics to measure performance. A common metric is the percent return over the trading period of a market-tracker like the S&P 500, which is effectively a weighted average of the performance of hundreds of publicly-trade companies. For trading individual stocks, tracking just the stock's performance over the trading period is analogous to such market-trackers. The baseline strategy that will be compared to the RL agent consists of buying the maximum amount of

shares at $t = 0$ and performing no other transactions throughout the trading period, which will be denoted LONG. This strategy's performance is then entirely dependent on the stock's performance, measured by total net worth computed at the end of the trading period, which is equivalent to total percent return if all agents begin with the same net worth.

3.3. Reinforcement Learning Algorithms

Advantage Actor-Critic (A2C). A2C is a variant on the Actor-Critic model, which is a hybrid between value-based and policy-based learning. At a high level, an "actor" decides on an action at each time step, and a "critic" provides feedback to the actor as to how optimal their action was. Both actor and critic then update their policy and value function, respectively, at the end of each time step, such that the actor takes better actions and the critic provides better feedback in future time steps. Two separate neural networks are maintained such that the policy function $\pi(s_t, a_t, \theta)$ parametrized by θ and value function $V(s_t, \omega)$ parametrized by ω can be updated at each step by the update equation

$$\nabla_\theta J(\theta) \sim \sum_{t=0}^{T-1} \nabla_\theta \log \pi(s_t, a_t, \theta) A(s_t, a_t)$$

where $A(s_t, a_t)$ is defined as the advantage function

$$A(s_t, a_t) = R_{t+1} + \gamma V(s_{t+1}, v) - V(s_t, v)$$

Above, γ is a constant rate and the advantage function measures the optimality of the action a_t given the state s_t compared to the average over all actions in the action set $V(s_t, v)$ parametrized by v. The advantage function stabilizes learning by reducing variability of reward values such that the computed gradients at each step are less noisy, improving time to convergence compared to standard policy gradient algorithms.

Proximal Policy Optimization (PPO). PPO is a variant on vanilla policy gradient methods that uses a clip function to prevent huge changes to the policy between steps. The standard objective

function to be optimized typically takes the form

$$L^{PG}(\theta) = \mathbf{E}_t[\log \pi(s_t, a_t, \theta) A(s_t, a_t)]$$

where $A(s_t, a_t)$ can be the advantage function described in the previous section. However, maximizing this objective function can result in dramatic policy updates between steps. Another objective function that can be used is

$$L^{CLIP}(\theta) = \mathbf{E}_t[\min(r_t(\theta)A(s_t, a_t), \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A(s_t, a_t))]$$

where $\varepsilon = 0.2$ and

$$r_t(\theta) = \frac{\pi(s_t, a_t, \theta)}{\pi(s_t, a_t, \theta_{\text{old}})}$$

which represents the ratio between the probability of the action a_t under the current policy and the probability of that same action under the previous policy. The use of the clip function and minimum function in conjunction with a different objective function is known as the Clipped Surrogate Objective and prevents destructively large updates to the policy [7].

Deep Q-Learning (DQN). Q-learning is a value-based learning algorithm that approximates and attempts to maximize the quality value, defined to be the cumulative reward over all time steps until termination. Formally, the quality value can be defined by the Bellman Equation

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a')$$

The equation defines a recursive relation where the quality value at time t in state s_t is equal to the reward received for doing action a_t plus the maximum quality value achievable from state s_{t+1}. Deep Q-Networks combine Q-learning with neural networks, which take as input s_t and output the quality values for each possible action where the action that maximizes the quality value is selected

at each step. Equivalently, the goal is to minimize the cost function

$$C(s_t, a_t) = [Q(s_t, a_t, \theta) - (r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a', \theta))]^2$$

where $Q(s_t, a_t, \theta)$ is parametrized by θ [4].

3.4. Financial Indicators

As previously described, defining the state space to represent the stock market proves more difficult than, for instance, defining the state for an Atari game (which typically consists of simply a flattened 1-dimensional array representing the RGB color value of each pixel). Defining the state space to just consist of stock prices is proposed in [9]. Using a combination of close prices and financial indicators such as moving average convergence/divergence (MACD) and relative strength index (RSI) to represent state space is proposed in [10]. As indicators such as MACD and RSI are commonly used as trade signals, this project will compare the performance of a state space consisting of just daily closing prices with a state space consisting of both daily closing prices and a financial indicator known as the histogram.

The n-day exponential moving average (EMA) of a stock is a weighted average of the stock's closing price from the last n days, with recent prices carrying more weight. The n-day EMA on day t is defined by the recursive relation

$$\text{EMA}_t(n) = \frac{2}{n+1} p_t + (1 - \frac{2}{n+1}) \text{EMA}_{t-1}(n)$$

for $t \geq 1$, where p_t is the closing price on day t and

$$\text{EMA}_0(n) = \frac{1}{n} \sum_{t=1-n}^{0} p_t$$

is defined to be the arithmetic mean of closing prices from $t = -(n-1) = 1-n$ to $t = 0$. The

MACD at day t is defined as the difference between the 12-day EMA and 26-day EMA, namely

$$\text{MACD}_t = \text{EMA}_t(12) - \text{EMA}_t(26)$$

for $t \geq 0$. The signal line (SL) is defined to be the 9-day EMA of the MACD series, namely

$$\text{SL}_t(0) = \frac{2}{n+1}\text{MACD}_t + (1 - \frac{2}{n+1})\text{SL}_{t-1}(n)$$

for $t \geq 1$ and

$$\text{SL}_0(n) = \frac{1}{n}\sum_{t=1-n}^{0} \text{MACD}_t$$

where $n = 9$ and SL_0 is defined similarly to EMA_0. Finally, the histogram on day t, which will be denoted HISTO_t, is defined to be the difference between the MACD and SL on day t, namely

$$\text{HISTO}_t = \text{MACD}_t - \text{SL}_t$$

for $t \geq 0$. The HISTO series is used frequently by stock traders but can be interpreted in several ways. Perhaps the most common interpretation strategy is *histogram reversal*, where if both $\text{HISTO}_{t-1} < 0$ and $\text{HISTO}_t > 0$, the market is interpreted as bullish and signals sell. Similarly, if both $\text{HISTO}_{t-1} > 0$ and $\text{HISTO}_t < 0$, the market is interpreted as bearish and signals buy. Furthermore, HISTO will increase as the market moves strongly in one direction and decrease as momentum slows [3].

4. Implementation

4.1. Data Processing & Libraries

Historical closing prices of MSFT, JNJ, WMT, DIS, and V from 2005 to 2019 are provided by Wharton Research Database Services. Historical closing prices of the S&P 500 are provided by Yahoo Finance. NumPy and Pandas are used for data manipulation. Microsoft Excel is used to

calculate the EMAs, MACD, SL and HISTO of each of the five stocks. The Open AI Gym library is used to build a custom stock trading Gym environment. The standard DQN, A2C, and PPO2 algorithms from the Stable Baselines library are used to train the agent with 25000 timesteps.

4.2. Action Space

The action space is the discrete set $A = \{0, 1, 2\}$, where 0 corresponds to buying the maximum amount of shares, 1 corresponds to sell, and 2 corresponds to hold. While a continuous action space A' could be considered to allow the agent to choose tuple actions that specify transaction *type* and transaction *amount* from the set $A' = \{0, 1, 2\} \cup [0, 1]$, where the first tuple value represents buy, sell, or hold and the second tuple value is a real number indicating the percent of the maximum transaction possible, this action space is not appropriate for a setting where the agent is trading only one company's stock. Intuitively, if the agent determines it is an optimal time to, say, sell some percentage α of its MSFT stock holdings at time t where $\alpha < 1$, it could certainly gain strictly more profit, and hence strictly more reward, by instead just selling all of its holdings where $\alpha = 1$. Thus, A is sufficient and effectively equivalent to A' and significantly simplifies the model. A continuous action space would perhaps be more appropriate within an asset allocation environment, where the agent must maintain a portfolio of multiple stocks. In this case, it certainly could be the case that, for instance, both buying some amount of MSFT stock and some amount of WMT stock on day t is optimal, and would require consideration of a more complex continuous action space or the union of several continuous spaces.

4.3. Environment

The agent starts with an initial wallet of $\text{WALLET}_0 = 10000$ and zero stock holdings represented by $\text{STOCKS}_0 = 0$. At time t, the agent a's net worth N_t is defined to be

$$N_t = \text{STOCKS}_t \cdot p_t + \text{WALLET}_t$$

where p_t is the closing price of the stock on day t. Thus, N_0 = 10000 for all agents. When the agent buys stock at time t, its WALLET and STOCKS are updated as follows:

$$\text{WALLET}_t = \text{WALLET}_{t-1} - \left\lfloor \frac{\text{WALLET}_{t-1}}{p_t} \right\rfloor$$

$$\text{STOCK}_t = \text{STOCK}_{t-1} + \left\lfloor \frac{\text{WALLET}_{t-1}}{p_t} \right\rfloor$$

where WALLET_t and WALLET_{t-1} are the agent's WALLET values on days t and $t-1$, respectively, and p_t is the stock's closing price on day t. Similarly, when the agent sells stock at time t, its WALLET and STOCKS are updated as follows:

$$\text{WALLET}_t = \text{WALLET}_{t-1} + p_t \cdot \text{STOCK}_{t-1}$$

$$\text{STOCK}_t = 0$$

When the agent holds, WALLET_t = WALLET_{t-1} and STOCK_t = STOCK_{t-1}. An episode terminates when either the agent's net worth becomes less than 0 or the current time step exceeds the end of the training period. The initial step is chosen randomly within the integer interval $t \in [75, 125]$ during training such that the agent can learn to optimize under a wide range of closing price history. The initial step is set to $t = 100$ during testing to allow consistent averaging over several episodes.

4.4. Reward Function

CURR-PREV. Three reward functions and their effects on all RL algorithms' performances are evaluated. The CURR-PREV reward function defines the incremental reward at each step

$$\text{CURR-PREV}(p_b, p_s, p_t, a_t, N_t, N_{t+1}) = \begin{cases} p_s - p_t & a_t = 0 \\ p_t - p_b & a_t = 1 \\ N_t - N_{t+1} & a_t = 2 \end{cases}$$

where p_s and p_b are the closing prices of the stock at the most recent sell transaction and buy transactions before day t, respectively, p_t is the closing price of the stock on day t, $a_t \in A$ is the action taken on day t, and N_t and N_{t+1} are the net worths of the agent on days t and $t-1$, respectively. Intuitively, CURR-PREV rewards and penalizes the agent directly proportional to its changes in net worth as a result of its actions at each step.

MIN-MAX. The MIN-MAX reward function encourages buys at stock valleys and sells at stock peaks by outputting a relatively high "bonus" reward whenever the agent makes an appropriate trade immediately following a local minimum or local maximum in the closing price series. MIN-MAX defines the incremental reward as

$$\text{MIN-MAX}(p_b, p_s, p_t, a_t, N_t, N_{t+1}, p_{t-1}, p_{t-2})$$

$$= \begin{cases} \text{CURR-PREV}(p_b, p_s, p_t, a_t, N_t, N_{t+1}) + \text{BONUS} \cdot \mathbf{1}(p_{t-1} < \min(p_t, p_{t-2})) & a_t = 0 \\ \text{CURR-PREV}(p_b, p_s, p_t, a_t, N_t, N_{t+1}) + \text{BONUS} \cdot \mathbf{1}(p_{t-1} > \max(p_t, p_{t-2})) & a_t = 1 \\ \text{CURR-PREV}(p_b, p_s, p_t, a_t, N_t, N_{t+1}) & a_t = 2 \end{cases}$$

where BONUS = 100 and $\mathbf{1}$ is an indicator that equals 1 when the input condition is satisfied and 0 otherwise. The indicator allows the agent to earn additional reward if it makes the appropriate transaction type one time step after a peak or valley, implying the agent can be delayed with respect to the market's movement by at most one day to earn the bonus reward.

GAME. The GAME reward function models stock trading like a survival task where total reward is proportional to time elapsed, like the Cart-Pole Problem [5]. For each day the agent's net worth stays equal to or greater than the LONG strategy's net worth, a step reward of 1 is awarded. The termination condition is modified such that a new condition is added: if the agent's net worth is less than the LONG strategy's net worth at some step, the episode ends and an end step reward of -10 is given. This incentivizes the agent to "survive" the episode for as long as possible to accumulate more reward.

4.5. State Space

5-STOCK. Three state spaces and their effects on the RL algorithms' performances are evaluated. The 5-STOCK state space s_t at time t consists of WALLET_{t-1}, N_{t-1}, the past 5 days of closing prices $p_{t-4} : p_t$, and an indicator that equals 1 if a local maximum occurred on day $t-1$, -1 if a local minimum occurred on day $t-1$, or 0 otherwise. s_t is thus represented as a 1-dimensional array of length 8.

5-HISTO. The 5-HISTO state space consists of all the same observations in 5-STOCK, with the addition of the past 5 days of HISTO values $\text{HISTO}_{t-4} : \text{HISTO}_t$, represented as a 1-dimensional array of length 13.

60-HISTO. The 60-HISTO state space consists of all the same observations in 5-HISTO, except instead of the past 5 days of closing prices and HISTO values, the past 60 days of closing prices $p_{t-59} : p_t$ and $\text{HISTO}_{t-59} : \text{HISTO}_t$ values are observed instead, represented as a 1-dimensional array of length 123.

4.6. Training & Testing Interval

3-TRAIN. The agent is trained on 2005-2016 closing price data, and tested on 2016-2019 data.

12-TRAIN. The agent is trained on 2016-2018 data and tested on 2019 data.

5. Evaluation

To evaluate the effect of the various training time intervals, reward functions, and state spaces, one component is variable while the other two are held constant. The optimal combination can then be determined and evaluated further. The three RL algorithms presented earlier are trained and tested against the LONG strategy, with the net worths at termination used to determine performance. All testing results are averaged over 100 episodes to ensure consistent results, and agents are run on all 5 individual stocks. The results indicate that the optimal combination is 3-TRAIN, 5-HISTO, and MIN-MAX, with up to 11% higher return on MSFT with the DQN algorithm.

5.1. Reward Function Performances

Tables 1-3 indicate that the DQN performs better on average on every stock using the MIN-MAX function than the CURR-PREV or GAME functions, averaging 2.75% better than the LONG strategy, a remarkable 11.08% better on MSFT, and 0.67% better on all stocks excluding MSFT. A2C and PPO2 don't perform as well as either LONG or DQN in most cases.

	LONG	DQN	A2C	PPO2
MSFT	12191.04	**13541.44**	11096.81	11684.52
JNJ	10432.23	**10451.01**	10228.20	10324.21
WMT	11420.46	**11511.55**	10722.42	11036.21
V	11329.36	**11399.56**	10624.76	10590.96
DIS	10484.36	**10598.17**	10244.11	10378.28

Table 1: Performance of 3-TRAIN, MIN-MAX, and 5-HISTO.

	LONG	DQN	A2C	PPO2
MSFT	**12168.15**	12150.26	12134.56	11861.49
JNJ	10459.78	**10460.34**	10430.74	10315.66
WMT	11432.63	10000.00	**11483.96**	11053.28
V	**11340.34**	11328.77	11320.80	11162.88
DIS	**10587.80**	10000.00	10001.83	10388.82

Table 2: Performance of 3-TRAIN, CURR-PREV, and 5-HISTO.

	LONG	DQN	A2C	PPO2
MSFT	9994.24	9963.43	**10028.60**	9939.57
JNJ	10008.21	9978.89	**10016.73**	9993.31
WMT	**10070.18**	10054.36	10053.32	10110.33
V	10098.93	10000.00	**10118.55**	10063.88
DIS	**10069.89**	10048.44	10053.48	10043.92

Table 3: Performance of 3-TRAIN, GAME, and 5-HISTO.

There may be several reasons why MIN-MAX produces the best result. One possible reason may be that MIN-MAX directly rewards the agent significantly for buying stock immediately following a valley and selling stock immediately following a peak, and penalizes the agent significantly for doing the opposite. Furthermore, the indicator in the state space allows the agent to directly associate a peak or valley with a sell or buy, respectively. The agent thus learns to follow the "buy low, sell high" doctrine while still being influenced by the CURR-PREV function on a smaller scale. As

mentioned before, the agent lags behind the market by one day, that is, instead of optimally buying or selling *at* a local minimum or local maximum, respectively, it performs the action one day later. Perhaps a better implementation of MIN-MAX is a function that outputs significant reward when the agent makes appropriate trades following histogram reversals, which it has access to within the state space. As described previously, HISTO tracks longer trends in the market and may be more indicative of optimal buy or sell times. This would allow the agent to confirm momentum in the market before making a transaction rather than attempt to take advantage of every local minimum or maximum as it does in MIN-MAX.

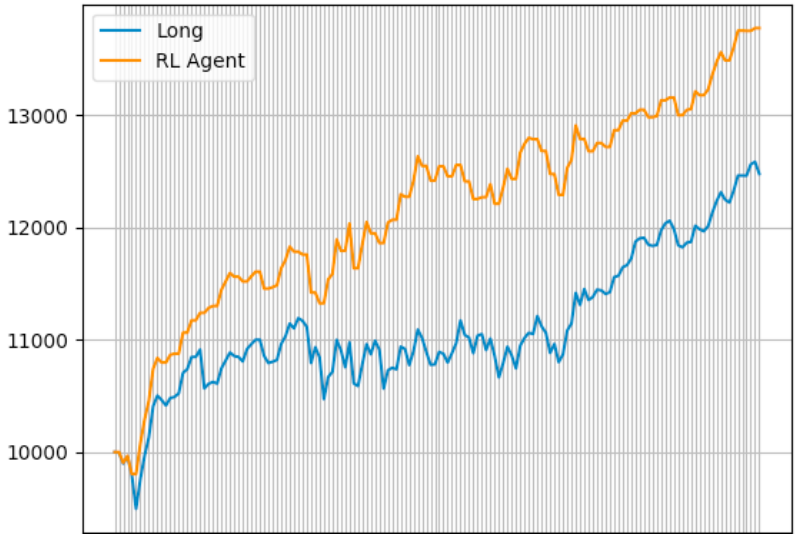

Figure 2: DQN with MIN-MAX vs. LONG averaged over 100 episodes on MSFT.

The CURR-PREV function by itself doesn't perform particularly well. The function ensures the agent learns to sell at a higher price than it previously bought and buy at a lower price than it previously sold. However, this prevents the agent from forecasting and waiting to make a transaction at a better time to maximize its reward and hence its total net worth. Rather, it learns to make a transaction as soon as it yields some positive reward that, more often than not, could have been larger had the agent chosen to hold longer. This holds particularly true when analyzing the equities market, which tends to trend upwards long-term. It is thus almost always the case that, immediately following a buy, holding for some time before identifying and selling at the next maximum is yields higher returns than simply selling at the first time the price is greater than the buy price.

MIN-MAX does precisely this: the agent attempts to identify peaks and valleys to maximize its reward long-term and thus trades less frequently as it learns that holding and riding an upwards trend will yield greater returns.

It is also worth noting that GAME produces final net worths that are very close to the initial net worth of 10000. It is safe to reason that this occurs because the episodes don't last very long due to the formulation of the GAME function, as early mistakes by the agent result in termination of an episode. Furthermore, the agent also frequently makes one good trade near the beginning of the trading period, after which it chooses to hold until termination. This isn't surprising as the GAME step reward is time-based and appropriately awards the agent for adopting such a strategy. As shown in Figure 3, the agent's long-term hold strategy yields a net worth almost identical to LONG's throughout the trading period, indicating little transaction activity.

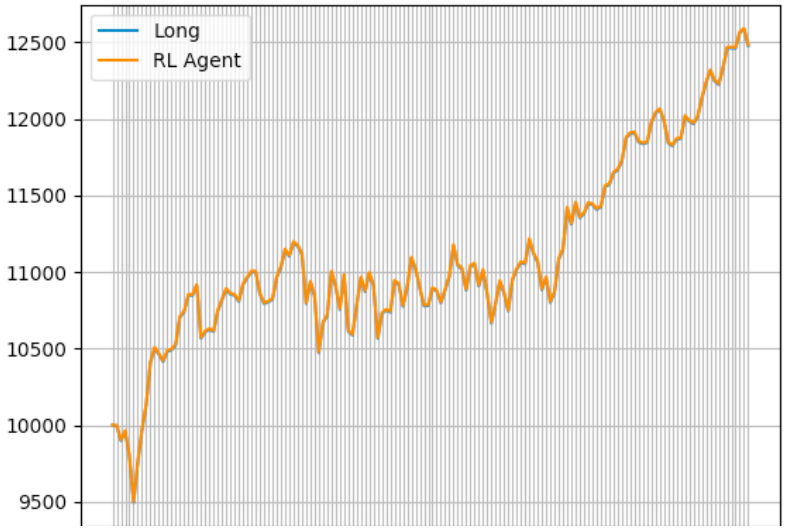

Figure 3: DQN with GAME vs. LONG averaged over 100 episodes on MSFT.

5.2. State Space Performances

Tables 1, 4, and 5 indicate that the DQN performs better on average on every stock using the 5-HISTO state space than the 5-STOCK or 60-HISTO state spaces. In every case, either LONG or DQN outperform A2C and PPO2. The 3-TRAIN, MIN-MAX, and 5-HISTO combination again proves to be best.

	LONG	DQN	A2C	PPO2
MSFT	**12223.67**	12207.43	11098.28	11498.44
JNJ	10423.62	**10472.25**	10208.98	10220.58
WMT	**11467.29**	10621.24	10658.44	10989.98
V	11265.54	**11327.05**	10624.91	10821.78
DIS	**10529.24**	10000.00	10349.39	10339.24

Table 4: Performance of 3-TRAIN, MIN-MAX, and 5-STOCK.

	LONG	DQN	A2C	PPO2
MSFT	**12206.36**	12188.52	11016.48	11564.06
JNJ	**10492.30**	10322.50	10256.55	10277.10
WMT	11481.46	**11653.13**	10729.71	10862.86
V	**11382.28**	11354.46	10739.60	11130.54
DIS	**10567.07**	10000.00	10355.40	10080.93

Table 5: Performance of 3-TRAIN, MIN-MAX, and 60-HISTO.

5-HISTO may perform well by providing the right amount of information needed for the agent to make decisions. In 5-STOCK, just closing prices might not indicate much about market movement. With a 5-day history of HISTO values in 5-HISTO, the agent is given a much better overarching view of momentum and can decide on an action accordingly, determining the estimated reward using closing prices and using both immediate and long-term data. 60-HISTO seems add too much noise in the state space, as closing prices and histogram values from 60 days ago may give little to no indication about market movement compared to the last 5 days, at least in the context of short-term trading with a trading period of less than 1 year. Rather than force the agent to associate a weight with an additional 55 days of older closing prices and HISTO values, it appears advantageous to reduce the state space and only present recent, more relevant data that better summarizes the market.

A potentially better state space would be comprised of more financial indicators such as RSI and EMA. Though various EMAs were computed to calculate HISTO, they aren't incorporated in the state space. Given that various technical indicators convey different information about market movement, including more could help the agent make better-informed actions.

5.3. Time & Testing Interval Performance

Tables 1 and 6 indicate that DQN performs better on average on every stock with 3-train than with 12-TRAIN. In every case, either LONG or DQN outperform A2C and PPO2. The 3-TRAIN, MIN-MAX, and 5-HISTO combination again proves to be best.

	LONG	**DQN**	**A2C**	**PPO2**
MSFT	22637.04	**22708.79**	15289.71	18946.77
JNJ	**11317.15**	11023.03	10703.28	10972.95
WMT	**15496.28**	12189.59	12755.10	12512.43
V	**19922.76**	15889.68	14395.80	17873.36
DIS	13216.98	**13271.78**	13250.22	13243.57

Table 6: Performance of 12-TRAIN, MIN-MAX, and 5-HISTO.

One possible reason as to why a longer training period results in worse performance is similar to why 60-HISTO counter-intuitively performs worse than 5-HISTO despite providing more data for the agent to learn from: 2005 market trends learned by the agent may not be relevant to 2016, especially since the testing interval is significantly shorter than the training interval. In 3-TRAIN, recent 2016-2018 data may be a much more relevant to the agent's performance in 2019. For instance, 12-TRAIN incorporates the 2008-2009 stock market crash, perhaps causing the agent to learn to expect volatility when the testing data is much less volatile and trends upwards. Meanwhile, the training and testing data presented in 3-TRAIN consistently demonstrates upward market trends.

6. Conclusion

6.1. Strengths

The two goals of this project were to train an agent that can outperform the LONG strategy and to develop an intuition as to what state spaces, action spaces, and reward functions yield the best returns in the context of several RL algorithms. 3-TRAIN, MIN-MAX, and 5-HISTO perform up to 11% better than the baseline LONG strategy on MSFT as shown in Figure 3, and an average of 2.75% better than LONG across all tested stocks. Clearly, RL agents can generate higher returns than the long strategy and beat the market. Intuitively, by incorporating a state space that includes

technical indicators like HISTO that describe market movement as well as a reward function that incentivizes the agent to trade at local minima and maxima, the agent learns to take advantage of market movement and buys at valleys while selling at peaks. This behavior is demonstrated in Figure 4, which plots the buy and sell actions of the DQN agent over time, where the x-axis is the time step and the y-axis is the price of MSFT.

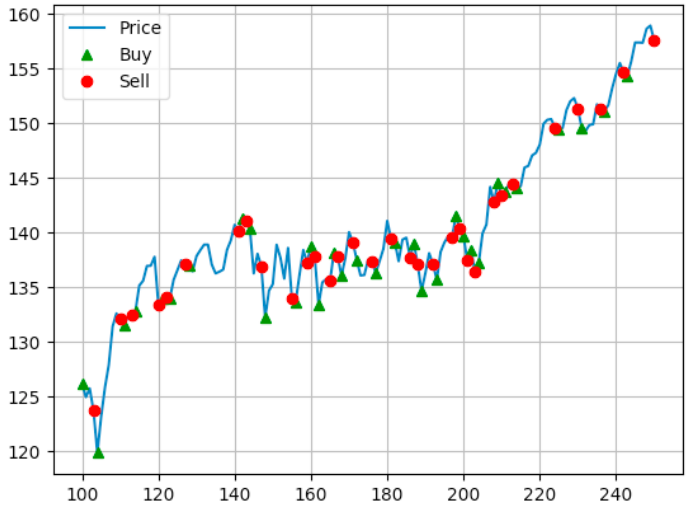

Figure 4: Transactions of DQN with 3-TRAIN, MIN-MAX, and 5-HISTO on MSFT.

Furthermore, we have introduced three reward functions that incentivize the agent in different ways. In particular, the GAME function models step reward similarly to a game where the total score is proportional to time. The MIN-MAX function models the reward in a more traditional sense by training the agent to identify peaks and valleys. In addition, the 5-HISTO state space balances market momentum data via technical indicators with little noise. The 60-HISTO state space may provide older, less relevant data not associated with present-day trading, which may be particularly noisy for short-term trading intervals. 3-TRAIN also proved to perform better than 12-TRAIN for similar reasons. In this way, we have surveyed a variety of environments and built a solid foundation for future work.

6.2. Limitations & Future Work

It is important to know the limitations of the project and possible extensions to this work. Overall, none of the models outperformed the LONG strategy by a wide margin on average. As mentioned in [10], this may be because equities markets demonstrate upward trends, with the occasional market crash. Thus, LONG is a great strategy that almost guarantees returns, especially with blue-chip stocks that are expected to demonstrate long-term, consistent growth. It is thus difficult to beat LONG by a significant margin unless the agent takes advantage of several valley-peaks, which tend to be temporary and small scale compared to overall growth. This implies LONG is a much safer strategy that achieves a large percentage of the maximum return over any trading interval in the context of equities, and may be preferred by risk-averse investors. This explains why a large percentage of retail investors choose to adopt the LONG strategy, particularly when they seek long-term gains.

The strictly quantitative approach in this work also doesn't paint the entire picture of stock market movement. For instance, even though the DQN agent does particularly well on MSFT, there are several reasonable explanations to this result that may not be encapsulated by just quantitative data like closing prices and technical indicators. In fact, the largest peaks and valleys can be explained almost entirely by qualitative data: the 2002 economic downturn caused largely by the September 11 attacks and Dot.com bubble burst, the 2008 financial crisis caused by the prevalence of subprime mortgages, and most recently, the recent record-breaking bear market caused by the outbreak of COVID-19. Future work should consider how both quantitative data and qualitative data, such as economic news, can be leveraged, as the market is also extremely dependent on factors such as current events like M&A activity that may not be easily captured with numbers.

Several extensions to this work should also be considered. In this project, agents were trained and tested on the same stocks, that is, trained only on MSFT and tested only on MSFT or trained only on JNJ and tested only on JNJ. It would be interesting to evaluate the results of an agent trained on, say, several stocks and tested on several stocks. The agent could potentially learn more generally, which may be advantageous for short-term trading. The results from this project could also be

applied to portfolio management, which would consist of an agent deciding not only what type of transaction to execute at each step, but also which stocks to perform the transaction on and the amount of transaction on each stock. As most stock traders invest in more than one stock, this scenario is more realistic and applicable. In addition, the performance of RL agents in other asset classes such as commodities or forex currencies should also be explored and may potentially yield better results than equities, as they aren't dominated by a strong upwards trend.

7. Acknowledgements

I would like to thank Professor Karthik Narasimhan and Xi Chen for the great seminar this semester. With no previous exposure to reinforcement learning or independent work, I was able to explore a field of finance that I've been interested in for quite some time while applying RL in novel ways, thanks to their help and feedback from the rest of the class.

8. Honor Code

This paper represents my own work in accordance with University regulations. -Victor Hua

References

[1] S. Bhatt, *MDP Diagram*, Mar 2018. [Online]. Available: https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

[2] S. Gibbs, "Alphazero ai beats champion chess program after teaching itself in four hours," *The Guardian*. [Online]. Available: https://www.theguardian.com/technology/2017/dec/07/alphazero-google-deepmind-ai-beats-champion-program-teaching-itself-to-play-four-hours

[3] A. Hayes, *Investopedia*, May 2020. [Online]. Available: https://www.investopedia.com/terms/m/macd.asp

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[5] S. Nagendra, N. Podila, R. Ugarakhod, and K. George, "Comparison of reinforcement learning algorithms applied to the cart-pole problem," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep 2017. [Online]. Available: http://dx.doi.org/10.1109/ICACCI.2017.8125811

[6] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement learning for optimized trade execution," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 673–680. [Online]. Available: https://doi.org/10.1145/1143844.1143929

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[8] Z. Tan, C. Quek, and P. Y. Cheng, "Stock trading with cycles: A financial application of anfis and reinforcement learning." Expert Systems with Applications, 2011, pp. 4741 – 4755. [Online]. Available: https://doi.org/10.1016/j.eswa.2010.09.001

[9] Z. Xiong, X.-Y. Liu, S. Zhong, H. Yang, and A. Walid, "Practical deep reinforcement learning approach for stock trading," 2018.

[10] Z. Zhang, S. Zohren, and S. Roberts, "Deep reinforcement learning for trading," 2019.