ACADEMIC SERVICES
Faculty Of Science & Engineering

Coursework Cover Sheet

GOVINDARAJ
Shamili

Unit Code 6G7V0016_2425_1F	**Unit Name** Knowledge Representation and Reasoning (6G7V0016_2425_1F)
Submission Deadline 12-Jan-2025 21:00 Revised deadline	**Assignment Description** 1CWK100 - 1 Report 100% (6G7V0016_2425_1F)
Submission ID X78a4cf35	**Student Number** 24834013

This work is a Group Submission ✓ Yes ☐ No	**Name of Unit Leader** Ismail Adeniran

Date Received (Office Use Only)

Generated 11-Jan-2025 03:47

Mark

S-X78A4CF35-00095706-C

MULTI STOCK TRADING USING REINFORCEMENT LEARNING TECHNIQUE

Shamili Govindaraj
24834013@stu.mmu.ac.uk

Jayanthaa Dhanabal
24843359@stu.mmu.ac.uk

Danancha Chanda Perera Illeperuma Arachchige Don
24851362@stu.mmu.ac.uk

Abstract- This research applies Reinforcement Learning (RL) Techniques with a well-defined Policies: Tabular Q learning Policy and Deep Q Learning Policy to optimize trading decisions in financial markets. The agent is built on the principle of Markov Decision Processes (MDPs), which is trained by combining reinforcement learning with DQNN to optimize trading actions: Buy, Sell, or Hold. Using historical data from multiple stocks, the model represents market states through features like stock prices, trading volume, and the Relative Strength Index (RSI), with rewards determined by the profit or loss resulting from its decisions to maximize cumulative reward. Tabular Q-Learning employs a discretized state space to update Q-values iteratively based on the Bellman equation, while Deep Q Learning utilizes DNN to approximate the Q-function, incorporating advanced techniques for stability. The training process incorporates an epsilon-greedy policy to equalize exploration and exploitation, creates a reward mechanism based on dynamic prices, and batch updates from a replay buffer for efficient learning. These results shows that the RL techniques, can be effectively applied to multi-stock trading as it provides a scalable solution for intelligent trading systems, enabling adaptation to diverse and dynamic market conditions. Future work could involve additional market features, like sentiment analysis or indicators, and extending the model to real-time trading scenarios would test its practical applicability and robustness in evolving stock markets.

I. INTRODUCTION

In current trend, Investors use auto trading technologies to make right investment decisions in the stock market. Traditional way of algorithm trading is done by human experts, based on the mathematical and stock theories. Lately, Investors decision making process is combined with ML techniques (i.e. supervised algorithms). With technology advancements, Reinforcement learning came into application. However, Deep Learning has proven effective in financial stock market, its role has been limited to predict market volatility, prices movement, and the upcoming trend, without offering a direct trading action such as buy/hold/sell of a specific number of stock shares. Alternatively, Deep Reinforcement Learning (DRL), stood out as a powerful technique, allowing agents to develop optimal behaviours by engaging directly with their environment through trial and error, without human intervention.

This project is based on the comparison of RL Models- TQL and DQL, capable of predicting and responding to factors such as market volatility, price fluctuations, and economic changes. To establish a foundation for decision making, DQL model integrates deep neural networks to

adapt to various market states. By integrating these techniques, the model adapts dynamically to various market conditions and demonstrates the potential of artificial intelligence in optimizing financial trading strategies.

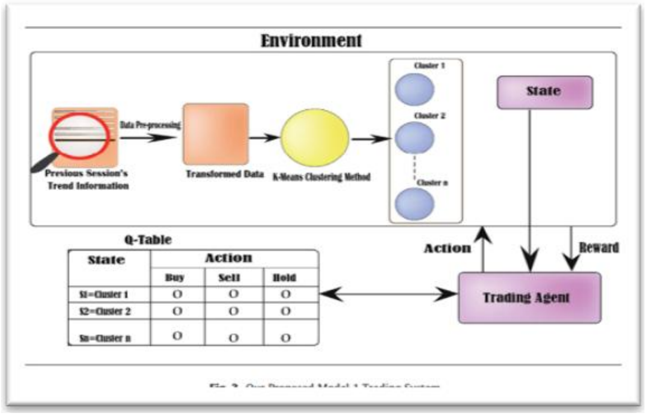
This project seeks to create an accurate and adaptive trading system that facilitates real-time trading decision (Buy/Sell/Hold), ultimately maximizing investment performance**.** Its highly relevant to institutional investors, hedge funds, financial technology companies, and retail investors seeking automated solutions to enhance their trading performance.

The report is structured to present the problem context and related research, followed by a detailed explanation of the methodology, results, discussion, Observation and recommendations for future improvements.

II. LITERATURE REVIEW

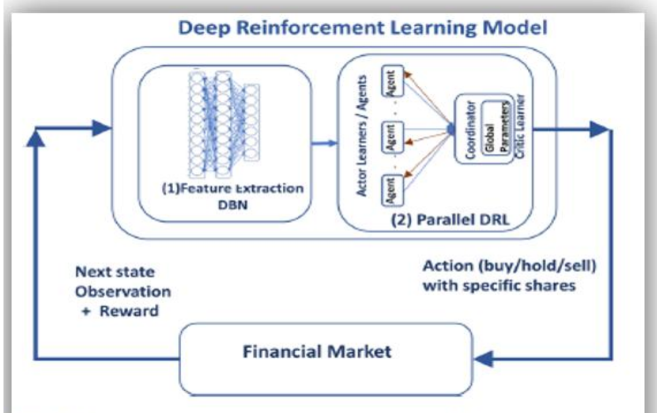
Firstly, most of the works in Algorithm trading are techniques, developed by traders and mathematicians without the use of AI. Lately, majority research focus on algorithms using ML for forecasting. Following this way, RL techniques have already been explored with good results. In research paper [2], its clearly defined how RL is capable to achieve optimality in dynamic algorithmic trading using price time series. The main focus of the paper is to improve the state functions for the RL model. A simple table was created for the RL stock trading system, utilizing the Q-learning algorithm to identify the optimal policy in dynamic market state. This innovation work has greatly impacted the field of RL in stock markets, establishing a conceptual foundation for future advancements in autonomous trading systems.

Progressing the literature study to next resource, the Q learning approach for algo trading is discussed in [3], proposes the development of a Q-Learning algorithm to train an agent to make trading decisions (Buy/Sell/Hold) in a real time trading environment. As shown in figure, the state spaces are made into cluster to depict the real time market environment, based on the generated Q table values the dynamic trading actions are rewarded.



State	Action		
	Buy	Sell	Hold
S1=Cluster 1	O	O	O
S2=Cluster 2	O	O	O
Sn=Cluster n	O	O	O

The proposed RL model outperformed traditional strategies in terms of profitability, demonstrating the potential of Q-learning for dynamic and adaptive decision-making in stock trading. This study facilitated the exploration of advanced RL algorithms, The paper [1] gave an insight on deep reinforcement learning using deep neural networks (DBN and LSTM) for predicting stock price and trading actions for multiple stocks in dynamic stock market conditions.



Deep Reinforcement Learning Model
(1)Feature Extraction DBN
Actor Learners / Agents
Agent
Agent
Agent
Coordinator
Global Parameters
Critic Learner
(2) Parallel DRL
Next state Observation + Reward
Action (buy/hold/sell) with specific shares
Financial Market

Fig. 3 Proposed deep reinforcement learning model architecture

The collective research from the literature study provides a holistic perspective on Reinforcement Learning (RL) in algo trading and the evolution and impact RL in finding optimal trading strategy.

III. METHODOLOGY

This section provides the brief overview of methodologies used in this project, includes Markov decision processes (MDSs) and RL based algorithms (Tabular Q and Deep Q-learning) as policies.

1. Markov Decision Process (MDP):

MDP is used to model a sequential decision-making problem under the condition of uncertainty. The MDP is a sequence of random states, with Markov property, where next state depends only on current state.

It defines the problem as a set (S,A,R,Z, γ) :
State (S)- Current condition of the system at the specific time, helps agent to make decisions.
Action (A)- Actions are the options available for the agent to take.
Reward (R)- Function that calculates the return (i.e. agents feedback)
Transition Functions(Z)- probability of moving from current state to next state
Discount factor (γ)- A value between 0 and 1, gives the significance of upcoming reward with respect to current reward.
The behaviour of agent to select action (Buy, Sell, Hold) based on current state, is maintained by the policy function:

$$\pi(a|s) = P(A_t = a|S_t = s)$$

The cumulative reward is calculated using:

$$X_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Where X_t depends on the series of actions taken, by this calculation the state action pair is defined. To evaluate the expected return from a given state-action pair within a policy, below formula is used:

$$Q_\pi(s, a) = E_\pi[X_t|s_t = s, a_t = a]$$

Thus, this MDP offers a best framework for RL based applications.

2. Reinforcement Learning Algorithms:

Generally, RL Consist of an agent and the environment as shown in figure [3] , to predict action and reward based on the discrete state function. The goal of agent is to maximize cumulative reward.

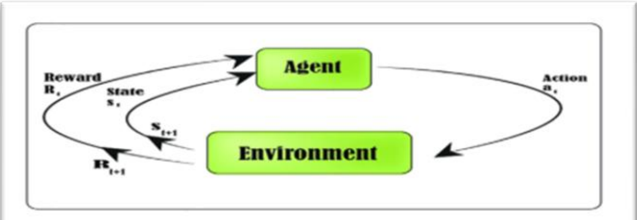


Fig. 1. Typical Reinforcement Learning system.

In this project RL algorithm is defined into policies functions. The policy gives the action to take for a given state, acting like a mapping function. They choose which action to implement from the options available, the one that affords the greatest reward. The policy is initially set to random, and selection becomes increasingly refined through experience of reward from the environment following the performance of activities in a given state.

2.a. Tabular Q Learning Algorithm:

The Q learning introduced by Watkins and Dayan in 1992, is a significant algorithm which doesn't delay on the model of the environment. One of the fundamental implementations of Q learning is Tabular Q learning approach, which works by updating the Q table that represents the Q state-action function Q (s_t, a_t)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

In particular, it uses the epsilon greedy strategy to build an ideal configuration for convergence of the Q-learning algorithm. Initially Q values are set to zero and it makes entries in a table, all of those correspond to a specified state and action. To update the Q-table at each iteration the exploration exploitation balance is used. During exploration a new state action pair is found, over time it shifts to exploitation, where it prioritizes actions that has higher Q-values based on the learned policy. This tailors the agent with the best decision-making framework to maximize long-term rewards. By using these optimal Q-values, the agent consistently selects actions that lead to the highest possible cumulative rewards in the given environment.

2.b. Deep Q neural network algorithm:

The DQNN is the value-based learning algorithm comprises of n-number of neurons which forms a deep neural network. In this technique the agent evaluates the value action pair and using greedy algorithm it chooses the most profitable expected reward, which defines it as a best decision-making algorithm in financial stock trading.

$$Q^*(s_t, a_t) = \mathbb{E}_{s_t,a_t}[r + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})|s_t, a_t]$$

This Q*(s_t, a_t) indicates the most optimal action (Buy, Sell, Hold) with a maximum reward value. By using the neural networks, the DQNN iteratively updates the action value function to yield the highest cumulative value.

The DQN minimizes a sequence of loss functions at each iteration t, to update the neural network weight θ_t:

$$L_t(\theta_t) = \mathbb{E}_{s_t,a_t}[(y_t - Q(s_t, a_t; \theta_t))^2]$$

Where y_t:

$$y_t = r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{t-1})$$

DQNN algorithm use techniques like experience replay buffer and mini bath updates to improve learning stability and efficiency.

IV. PROPOSED SYSTEM

This section explains the proposed model of our research. The rationale for using the RL based approach to derive the optimal dynamic trading strategy, based on the fact that sequential decision-making problems are modelled in the framework of MDPs are solved by reinforcement learning.

A. Main Functions:

1. State:

The state includes the various factors including prices-open, high, close, low and RSI. These features capture market dynamics and influence trading decisions. The Close Price is major for reward calculation, while RSI indicates overbought or oversold conditions.

2. Action:

The action in the trading environment consists of three key conditions: Buy, Sell, Hold. The Buy action involves purchasing the stock, while sell action involves selling the stock, in price decrease condition. The hold action indicates no trading. These actions allow the agent to respond to the market conditions dynamically.

3.Reward:

The reward is calculated by state and action, which is the important tool for training the model. This objective function makes sure that agent develops the decision-making skill aligned with the profit-making strategies. The formula to calculate reward is

$$Reward = \begin{cases} \text{Next Price} - \text{Current Price}, & \text{if action is Buy} \\ \text{Current Price} - \text{Next Price}, & \text{if action is Sell} \\ 0, & \text{if action is Hold} \end{cases}$$

Here, if buy action is made the reward obtained is the difference between Next price and current price. This happens inversely if sell action is made. The hold action reward is 0 indicating no profit or loss.

4.Exogeneous Information:

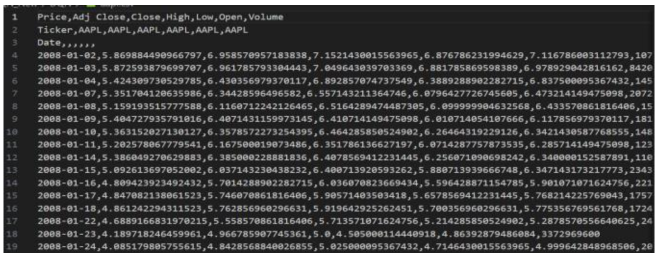
The exogenous elements in the trading environment include the market data (OHLCV) and the price transitions, such as the movement from current price to next price, are entirely driven by market dynamics and serve as key inputs for reward calculation.

5. Transition Function: After getting each reward, based on the action the state function is transitioned (i.e. the conversion of current price to next price with a different portfolio value.)

B. Data Loading and Training:

1. Pre-processing:

Market behaviours are effectively captured in the given stock market dataset, which includes data from January 2008 to December 2022 for multiple stocks: Google (GOOGL), Apple (AAPL), and Microsoft (MSFT), from the source "Yahoo Finance", which has features like open/close prices, trading volume, and adjusted closing prices. Related information is embedded within the dataset through sequential data points representing daily stock activity, reflecting price fluctuations and market trends. The below figure is the sample representation of the data set.



```
Price,Adj Close,Close,High,Low,Open,Volume
Ticker,AAPL,AAPL,AAPL,AAPL,AAPL,AAPL
Date,,,,,,
2008-01-02,5.869884490966797,6.958570957183838,7.1521430015563965,6.876786231994629,7.116786003112793,107
2008-01-03,5.872593879699707,6.961785793304443,7.049643039703369,6.881785869598389,6.978929042816162,8420
2008-01-04,5.424309730529785,6.430356979370117,6.892857074737549,6.3889288902282715,6.837500095367432,145
2008-01-07,5.351704120635986,6.34428596496582,6.557143211364746,6.0796427726745605,6.473214149475098,2072
2008-01-08,5.159193515777588,6.1160712242126465,6.5164289474487305,6.099999904632568,6.433570861816406,15
2008-01-09,5.404727935791016,6.4071431159973145,6.410714149475098,6.010714054107666,6.117856979370117,181
2008-01-10,5.363152027130127,6.3578572273254395,6.464285850524902,6.26464319229126,6.3421430587768555,148
2008-01-11,5.202578067779541,6.167500019073486,6.351786136627197,6.0714287757873535,6.285714149475098,123
2008-01-14,5.386049270629883,6.385000228881836,6.4078569412231445,6.256071090698242,6.340000152587891,110
2008-01-15,5.092613697052002,6.037143230438232,6.400713920593262,5.880713939666748,6.347143173217773,2343
2008-01-16,4.809423923492432,5.7014288902282715,6.036070823669434,5.596428871154785,5.901071071624756,221
2008-01-17,4.847082138061523,5.746070861816406,5.90571403503418,5.6578569412231445,5.768214225769043,1757
2008-01-18,4.861242294311523,5.762856960296631,5.919642925262451,5.700356960296631,5.775356769561768,1724
2008-01-22,4.6889166831970215,5.558570861816406,5.713571071624756,5.214285850524902,5.2878570556640625,24
2008-01-23,4.189718246459961,4.966785907745361,5.0,4.505000114440918,4.86392879486084,3372969600
2008-01-24,4.085179805755615,4.8428568840026855,5.025000095367432,4.7146430015563965,4.999642848968506,20
```

This comprehensive representation allows the RL model to analyse dynamic market behaviours and adapt its trading strategies accordingly. To simplify the RL process, the raw stock market data undergoes preprocessing, it involves converting columns to numeric, removing missing values, and normalizing features using MinMaxScaler to scale all values between 0 and 1. The RSI is calculated over a rolling window to capture recent price gains and losses, providing critical information on market momentum. The formula given below calculates RSI:

$$RSI = 100 - \frac{100}{1 + RS}, \quad RS = \frac{\text{Average Gain}}{\text{Average Loss}}$$

These steps ensure data consistency, improve training efficiency, and provide a robust foundation for the model to learn and adapt effectively to dynamic financial markets.

2. Data Training:

The training method involves applying two RL methods: Tabular Q learning (TQL) and Deep Q Network (DQN), to build a better trading strategy. In the case of TQL, the updating of Q table for each stock according to the given state features is done in iterations while the rewards used are logged to a performance metric. Similarly, DQN utilizes deep neural network to approximate Q values; each of the stock datasets is used to train a different DQN model. These trained models are used under the policy to evaluate and analyse the better trading decisions.

i. Training Process of TQL-Model: The agent iterates through the stock data over multiple episodes. The actions are selected for each episodes using the epsilon-greedy policy, and using the Bellman equation Q-values are updated. Rewards are accumulated, to improve agent's policy and maximize cumulative rewards over time.

ii. Training Process of DQN-Model: The DQN model is trained using states, actions, and rewards derived from stock market data over multiple episodes. Selecting Actions using epsilon-greedy policy, where € decays over time according to the below formula:

$$\epsilon = \max(\epsilon_{\text{min}}, \epsilon \cdot \epsilon_{\text{decay}})$$

During training, the Q-network updates its parameters using the Temporal Difference (TD) Loss, while a target network is periodically synchronized to stabilize learning.

C. Policies:

The trained model operates under two distinct policies with Reinforcement Learning algorithms:

1. Tabular Q Learning Based Policy
2. Deep Q Neural Network Policy

Policy 1: Tabular Q learning Policy:

This section describes the implementation of Q learning technique. The algorithm functions in the manner which the state space is divided into bins and then it is possible to establish a Q-table in which each entry is associated with a given state-action. First, the value function is updated in the Q table using the Bellman equation. Process includes:

State discretization: Continuous states are discretised into bins, thus making it easier for tabular representation of the state space.

Exploration and Exploitation plan: An action is chosen randomly or based on the highest Q value regulated by epsilon €-greedy policy. The formula is given below:

$$a = \begin{cases} \text{random action}, & \text{if } \epsilon > \text{random number in } [0,1] \\ \arg\max_a Q(s,a), & \text{otherwise.} \end{cases}$$

Reward Calculation and Q value Update: By this way, the reward function reflects trading actions, making it natural that its estimated values should guide the search for improvements to trading actions.

1.Positve price change triggers "Buy" reward (1)

2.The "Sell" reward when price drops (1)

3. "Hold" action yields no rewards (0)

Based on the reward function the algorithm updates the table Q value, generates episodes of successful decisions. At the end of each episode the cumulative reward is calculated to measure the performance of the policy.

Policy 2: Deep Q Neural Network Policy:

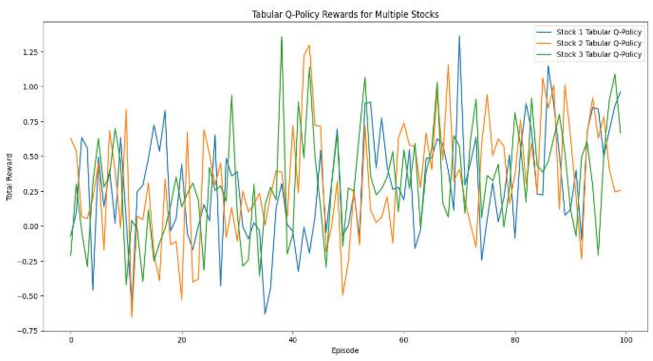
This section, follows the DQNN policy, an improved version of Q learning, which uses neural networks to estimate Q values of a continuous state-action space. The DQNN Architecture estimates the action-value function, accordingly the weights are adjusted in the learning phase. Another target network strengthens learning by eliminating temporal couplings.

Secondly, a Replay Buffer is initialized with a capacity of 10,000 stores past experiences, allowing random sampling during training. This helps break temporal correlations, ensuring the stable and efficient learning by reusing historical market data.

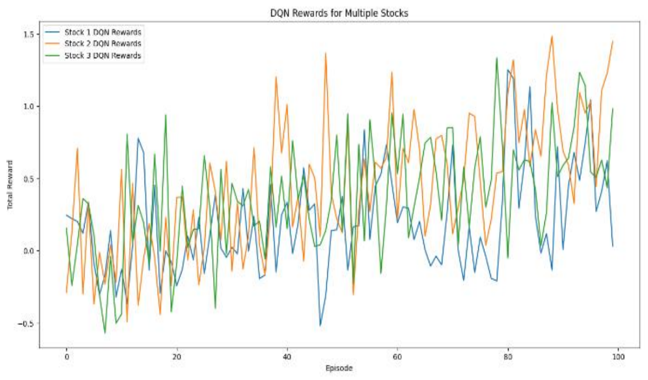
Making use of ϵ-greedy strategy, the exploration & exploitation action is performed. However, the Q-network is periodically copied to the target network to make learning stable during training process. In the process of experimentation, the algorithm records overall rewards by episode in judging the policy.

V. VISUALIZATION

POLICY 1: The graph shows the cumulative rewards obtained by the Tabular Q-Learning algorithm for multi-stocks over multiple episodes. The fluctuation shows the agent's exploration and learning process, with stabilization indicating adaptation to the trading environment of each stock.



POLICY 2: Cumulative rewards are plotted against the episodes to showcase the stabilization indicating adaptation to the trading environment of each stock. The fluctuation shows the agent's exploration and learning process.



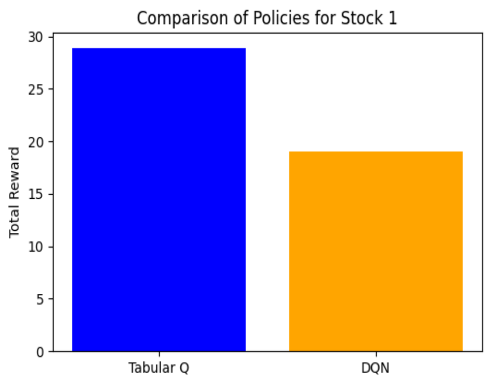
VI. RESULTS

The implementation of TQN and DQN Policy, across multiple stock datasets (AAPL, GOOGL, MSFT) shows varied reward patterns, with DQN demonstrating higher performance for some stocks. This difference is attributed to the DQN's ability to approximate Q-values using DNN, which is advantageous in handling the complexities of multi-stock environments. Different hyperparameters were tested, these including learning rate of 0.001, discount factor (γ) of at least 0.99 and epsilon decay of at least 0.995. Hyperparameters tuning was also extended to pursue enhanced performance with other hyperparameter deviations.

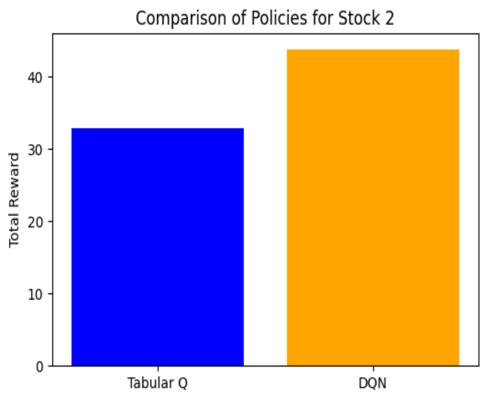
VII. OBSERVATION

The observations, presented in bar graphs for cumulative reward comparisons, showed that DQN achieved a higher total reward than Tabular Q-learning. However, for Stock 1, Tabular Q-learning performed slightly better, suggesting that the simpler algorithm can still be effective in specific scenarios. The result shows that Google (stocks 1) has a profitable result with tabular q policy, whereas the apple (stocks 2) and Microsoft (stocks 3) shows good results with DQNN policy, shows the stock values are more dynamic and less predictable, which needed effective algorithms like DNN to outperform.

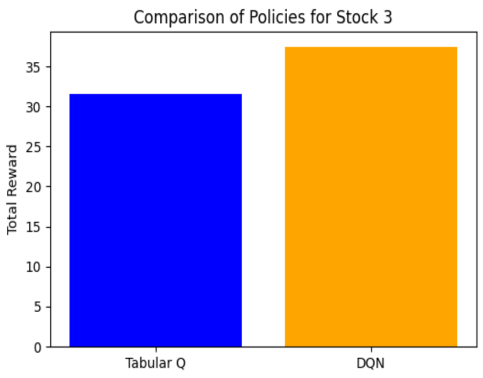
For Google stock:



For Apple stock:



For Microsoft Stock:



VIII. CONCLUSION

In conclusion, this project proposes the Multiple stock trading approach, incorporating RL algorithms. The results affirm the potential of DQN in adapting to complex stock environments, while also highlighting the value of Tabular Q-learning in certain cases. Through this analysis, it was evident that the choice of hyperparameters and the reinforcement learning algorithm plays a significant role in optimizing multi-trading strategies. The comparison highlights the basic approach and scaling by using RL in trading where the trade-off is achieved between simplicity and scalability. The future work could be directed to the hybrid methods or to real time data integration for even better performance.

IX. TEAM CONTRIBUTION

All three of us contributed equally in the report, separated between each of us and contributed as sections.

In the coding section:

1. Shamili- contributed on designing the main functions and trained both the RL models using the stock data.

2. Jayanthaa- Developed the TQL and DNN Model policies and generated cumulative results.

3. Danancha- Handled pre-processing and done analysis of graph for stock observations.

X. REFERENCES

[1] AbdelKawy, R., Abdelmoez, W.M. and Shoukry, A., 2021. A synchronous deep reinforcement learning model for automated multi-stock trading. *Progress in Artificial Intelligence*, *10*(1), pp.83-97.

[2] Aloud, M.E. and Alkhamees, N., 2021. Intelligent algorithmic trading strategy using reinforcement learning and directional change. *IEEE Access*, *9*, pp.114659-114671.

[3] Chakole, J.B., Kolhe, M.S., Mahapurush, G.D., Yadav, A. and Kurhekar, M.P., 2021. A Q-learning agent for automated trading in equity stock markets. *Expert Systems with Applications*, *163*, p.113761.

XI. APPENDIX

The GitHub repository for our Algorithm: code here

Knowledge Representation and Reasoning 6G67V0016
Peer Assessment for Groupwork (30%)

Peer review form for Group: Shamili- Jayanthaa- Danancha
Name of Reviewer: Shamili Govindaraj

Name of group member	CO-OPERATION /5	COMMUNICATION I /5	COMMUNICATION II /5	ORGANISATION /5	CONTRIBUTION /5	REFLECTION /5
Jayanthaa Dhanabal	5	5	5	5	5	5
Danancha Chanda Perera Illeperuma Arachchige Don	5	5	5	5	5	5