Study on Stock Market Trading and Strategy Development for EV Stocks Using DQN and Double-DQN Reinforcement Q-Learning Models

CSC-792 Reinforcement Learning

Alisha Karna
alisha.karna@coyotes.usd.edu

Madhavi Awasthi
madhavi.awasthi@coyotes.usd.edu

Pramesh Baral
pramesh.baral@coyotes.usd.edu

Pramod Acharya
pramod.acharya@coyotes.usd.edu

Randhir Kumar Yadav
randhirkumar.yadav@coyotes.usd.edu

Abstract

This study aims to investigate the effectiveness of deep reinforcement learning algorithms, specifically DQN and Double-DQN, in developing trading strategies for electric vehicle (EV) stocks, which are \$TSLA, \$GM, \$BYDDY, and \$NIO in the stock market listed in NYSE. The data sets are taken between the period of COVID-19 and economic inflation, 2022. The study utilizes historical stock price data of major EV companies and implements the DQN and Double-DQN algorithms to learn and predict optimal trading actions. This study suggests a technique for utilizing reinforcement learning, which is appropriate for simulating and comprehending diverse types of real-life interactions, to tackle the challenge of predicting stock prices. The performance of the proposed strategy is compared with the traditional buy-and-hold strategy, as well as with other popular trading strategies in the literature. The experimental results demonstrate that the DQN and Double-DQN algorithms can effectively learn and develop profitable trading strategies for EV stocks, outperforming the traditional buy-and-hold strategy and other benchmark trading strategies in terms of returns and risk-adjusted-performance. The findings suggest that deep reinforcement learning algorithms hold promise for developing effective trading strategies in the stock market, particularly for emerging sectors such as the EV industry.

Keyword: DQN, Double-DQN, Rewards, Q-value

1. Introduction

Reinforcement learning (RL) is a machine learning technique that enables agents to learn from their experiences in each environment through trial and error [1-5]. When applied to stock trading, RL agents can be trained to optimize buying and selling actions for maximum profits. The use of deep reinforcement learning in analyzing stock data has recently gained attention due to its potential to develop effective trading strategies [6,7]. The combination of RL and deep learning can provide a cognitive decision-making system that is well-suited for complex systems, especially in financial markets. This approach can handle large-scale financial data, improve data processing capabilities, and enhance transaction performance by identifying the most advantageous price in each situation.

In this study, we describe a deep reinforcement learning model for stock trading that employs the use of the capabilities of neural networks to identify intricate patterns in stock data and make wise trading judgments. To train an agent that can learn from its interactions with the stock market environment and execute behaviors that maximize long-term gains, we specifically combine deep learning and reinforcement learning methodologies. We put the model into practice using the Chainer deep learning framework, and we tested its effectiveness using a number of stock datasets. To evaluate the behavior of the model and acquire understanding of the underlying trends in the data, we also give visualization tools utilizing the Plotly library. Overall, our findings show the difficulties and possibilities for further research in this area, as well as the potential of deep reinforcement learning for stock trading. The Bellman equation expresses the relationship between the value of a current state, and the expected value of the next state, taking into account the reward obtained and the discount factor applied. It can be written as:

$$\underset{\text{optimal Q function for current step}}{Q^*(s_t, a_t)} = r_t + \underset{\text{truncate last Q value}}{\gamma(1-d)} \underset{\text{optimal Q function for next step}}{\max_a Q(s_{t+1}, a_{t+1})}$$

γ is the discount factor, which represents the importance of future rewards compared to immediate rewards.

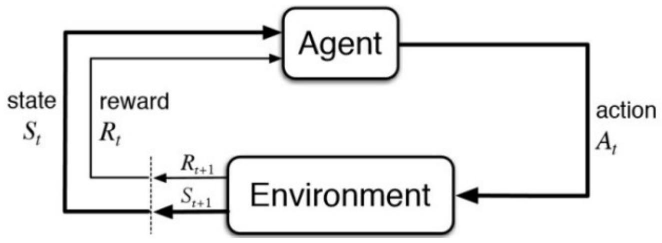


Fig 1: Flowchart of reinforcement learning

Deep neural networks (DNNs) excel at deriving lower-dimensional representations from high-dimensional input data, but they lack decision-making capabilities [7]. To solve this, a cognitive decision-making system can be developed by combining Deep Learning and Reinforcement Learning (RL). This method can manage enormous amounts of data, expand data processing capabilities, and improve transaction performance by determining the optimum price in the financial market. With the help of Q-learning and neural networks, the Deep Q-Network (DQN), a well-liked RL technique, selects actions that maximize a cumulative reward signal over time. DQN uses a target network to estimate target Q-values and experience replay to improve the learning process' stability. DQN employs an epsilon-greedy policy to strike a balance between exploration and exploitation. DQN can be utilized in this way to forecast stock market trading and learn a trading technique.

The Double Deep Q-Network (Double DQN) approach in reinforcement learning solves the issue of overestimation of Q-values in the Deep Q-Network (DQN) algorithm [7]. To lessen overestimation, it employs two different neural networks: a Q-network for action selection and a target network for Q-value evaluation. The next state's action is chosen by the target network, and the Q-network assesses its Q-value to update the Q-network. Additionally, Double DQN employs an epsilon-greedy policy and experience replay. It is a well-liked strategy for lowering overestimation of Q-value in reinforcement learning.

The Q-value for DQN model mathematically can be represented as,

$$Q(s,a) = r + \gamma \times \max(Q(s',a')).$$

Similarly, for Double DQN model can be expressed as,

$$Q(s,a) = r + \gamma \times Q(s', \arg\max(Q(s',a';\theta)); \theta^{-})$$

Where, Q(s,a) represents the Q-value of a state-action pair (s, a), r represents reward obtained by taking an action **an** in state **s,** s' represents the next state reached after taking an action **an** in state **s,** a' represents the action that maximizes the Q-value in state s', γ is the discount factor which determines the importance of future rewards, max(Q(s',a')) represents the maximum Q-value over all actions **a'** in state **s',** θ represents the parameters of the Q-network, Q(s,a;θ) represents the predicted Q-value for state-action pair (s,a), max(Q(s',a';θ^{-})) represents the maximum Q-value over all actions a' in state s' using a separate target network with frozen parameters θ^{-}, θ^{-} represents frozen parameters, refer to the weights and biases of a neural network that are fixed and do not get updated during the training process, argmax(Q(s',a';θ);θ^{-}) represents the action that maximizes the Q-value in state s' using a separate target network with frozen parameters θ^{-}.
The main advantage of Double Deep Q-Network (Double DQN) over Deep Q-Network (DQN) is that it addresses the overestimation issue present in DQN. In DQN, the maximum Q-value for the next state is selected using the same Q-network for both action selection and Q-value estimation. This can lead to overestimation of Q-values, which can result in unstable and sub optimal policies. Double DQN, on the other hand, uses two separate neural networks - one for action selection and the other for Q-value estimation. This technique reduces the overestimation issue, resulting in more accurate Q-value estimation and improved performance. Specifically, Double DQN provides a better estimate of the true value of the action by using the current Q-network to select the action and the target Q-network to estimate its value. This approach results in a more accurate approximation of the optimal action-value function, leading to better performance in reinforcement learning tasks.

2. Motivation

The motivation behind studying stock market trading and strategy development for EV (Electric Vehicle) stocks using DQN (Deep Q-Network) and Double-DQN reinforcement Q-learning algorithms is driven by the increasing demand for sustainable transportation solutions and the subsequent growth of the EV market [8,9]. As more and more investors are attracted to the EV industry, it becomes increasingly important to develop effective trading strategies that can help them make informed decisions in this highly dynamic and complex market. Here, we discuss the challenges of predicting stock prices and the limitations of traditional methods such as technical and fundamental analysis. It highlights the potential of machine learning techniques, particularly Reinforcement Learning, in stock market forecasting. RL algorithms learn from historical data and can identify patterns and trends to make predictions about future stock prices. In many areas, including stock market forecasting, Deep Q-Networks (DQN) and Double DQN application in Reinforcement Learning (RL) has shown promising results [10-12]. These deep learning architectures can result in better performance on RL problems by lowering the overestimation of Q-values. To anticipate the stock market, this research article will examine the efficacy of integrating Reinforcement Q-Learning with DQN and Double DQN. The study will assess the effect of various hyperparameters and training methods on its accuracy and efficiency as well as compare the performance of this approach to conventional methods like technical and fundamental analysis.

3. Literature Review

To forecast and trade the stock market, numerous methods have used deep reinforcement learning techniques, such as a DQN. Liang, Chen, Zhu, Jiang, and Li (2018) trained in DQN risk hedging for portfolios. A Deep Learning system is trained by Ding, Zhang, Liu, and Duan (2015) for event-driven stock prediction. Additionally, Li, Jiang, Li, Chen, and Wu (2015) use neural networks to forecast the stock market. This work improves on these earlier efforts by decorrelating training samples, reducing mistakes, and achieving improved performance using DQN and Double Deep Q-Network (DDQN). It differs from earlier efforts in that it extracts features with pairs trading, a particular trading technique, in mind. The efforts discussed earlier take advantage of input features like the portfolio assets themselves or news feed data. The system can learn at what time interval the spread mean reverts since in this case the input features represent the spread mean for various time lags. After that, the algorithm generates steps to forecast the spread. The DDQN interacts with an RL environment by taking positions on the spread of the stocks that are long, short, or neutral. The DDQN optimizes a Q-function that produces the best action for each given state of input features as it acts in the environment and earns rewards for each action.

4. Data Collection and Processing

The data used for the study was taken from Yahoo Finance. We have taken 4 different electric vehicle stocks that are listed on the New York Stock Exchange (NYSE), which are Tesla ($TSLA), General Motor ($GM), Nio ($NIO), and BYD Company ($BYDDY). Data includes the day-to-day opening and closing price with the highest and lowest value per unit stock. The data for stocks covers the period from Jan.02,2020 to Dec 29,2022. The data was cleaned and pre-processed after collection.

Figure 3 and 4 illustrates our attempt to explain the deployment's structure, which improves the efficiency, effectiveness, and precision of our study. When we receive raw data, we first go through a phase of data analysis to spot any extreme data while also processing the raw data itself. The Deep Q Network (DQN) segment is then filled with the processed data. A DQN is a sort of network that continuously updates the neural network to learn the maximum Q value while using a neural network to anticipate the Q value. The DQN consists of two neural networks: the Target-Network, which is used to acquire the target value and has comparatively set parameters, and the Current Q-Network, which is used to assess the current Q value. The Replay Memory, which retains the actions (a), rewards (r), and results of the following state (s, a, r, s'), is used to randomly extract training data. The Networks alter their settings in response to changes in the environment, and the Replay Memory does the same. The Q value in the Target-Network is subtracted from the Q value in the Current Q-Network to get the Loss function. Iterative adjustments are made to the values between modules until the desired Q value is obtained, at which point the output operation is carried out. The Double DQN approach, which is covered in more detail in the following section, has also been built using a similar process.

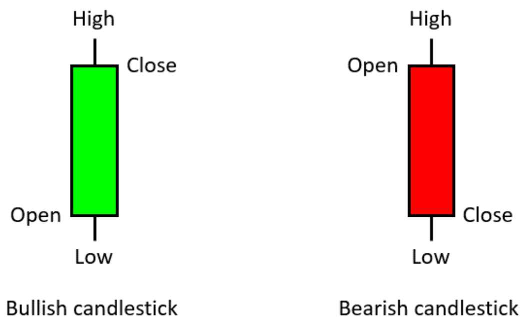


Fig 2: Candle stick in stock trading. A candlestick shows the highest and lowest price of a stock in a period, as well as the opening and closing price of the stock

5. Methodology

The experimental setup and its requirement were dependent on Python 3.10 version based on Tensor flow, OpenAI Gym, and Keras-RL deep-learning frameworks. We used Jupyter Notebook for the programming. The source code was derived from authentic web, and we converted it into our own required format for the implementation and study. The study focused on evaluating the performance of two popular Deep Reinforcement Learning models, namely Deep Q-Network (DQN) and Double Deep Q-Network (DDQN), through computational analysis and performance comparison. The study involved splitting each stock into a training and testing set, which were then used as input for the DRL models. By simulating trading and contrasting the advantages of the two models, the impact of the models was assessed. The study's conclusions were derived through a concurrent analysis of the training and test data.

5.1 Deep Q-Network (DQN:

Deep Q-Network (DQN), a popular reinforcement learning technique that blends deep neural networks and Q-learning, goes by the name Deep Q-Network [13-15]. An agent can learn a policy that maximizes the predicted cumulative reward by using the algorithm, which is built to approximate the ideal action-value function in a certain environment. DQN can be used in stock market prediction by training an agent to make trading decisions based on historical price and volume data. The agent can learn to maximize its expected cumulative reward by buying and selling stocks at the right time. In DQN, the value function is estimated using a single neural network, which takes the state as input and outputs the expected reward for each action. During training, the network is updated using the Bellman equation, which involves computing the maximum Q-value for the next state and using it to update the Q-value of the current state-action pair.

To be clear, utilizing DQN or any other machine learning algorithm for stock market prediction is not a surefire strategy to generate money because the stock market is a very complicated and unpredictable environment. Any prediction model's performance will be influenced by a number of variables, such as the data's quality, the characteristics used, and the trading strategy employed. DQN or any other machine learning method should therefore be used with caution for stock market predictions.

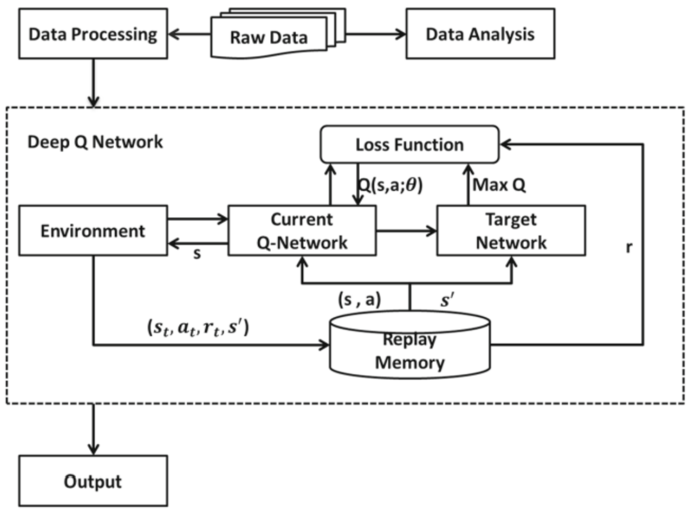


Fig 3: The deployment of DQN model on stock trading for data forecasting

5.2 Double Deep Q-Network (DDQN):

DDQN (Double Deep Q-Network) is a reinforcement learning algorithm that builds on the DQN algorithm to address the issue of overestimation of Q-values [13-15]. It achieves this by using a target network to estimate the maximum Q-value of the next state, while still using the online network to estimate the Q-values of the next state. This helps to reduce overestimation and improve the stability of the learning algorithm. Two neural networks are employed in Double DQN to estimate the value function.

The best action is chosen using one network, and the Q-value of that action is estimated using the other network. Specifically in situations when the target and evaluation networks have different estimations of the maximum Q-value, the goal is to lessen the overestimation of Q-values that can happen in DQN. The update algorithm for Double DQN has also been changed so that the evaluation network is used to estimate the Q-value and the target network is used to choose the optimum course of action.

The DDQN algorithm can be applied to the stock market by learning a profitable trading approach from it. To do this, the stock market can be modeled as a setting that incorporates the status of the market at any given time and delivers the reward for the agent's chosen action (buy, sell, or hold). Features like historical stock prices, trade volume, and other technical indicators can be used to describe the status. Using two neural networks, one new and one old, with the same structure but different internal parameters that are updated using time difference is known as double Deep Q learning. Since the neural network's prediction of the optimal Q value is inherently inaccurate, the error can grow with each iteration. Double DQN adds an additional neural network to help solve this issue. Together, the main network and the target network reduce the number of members.

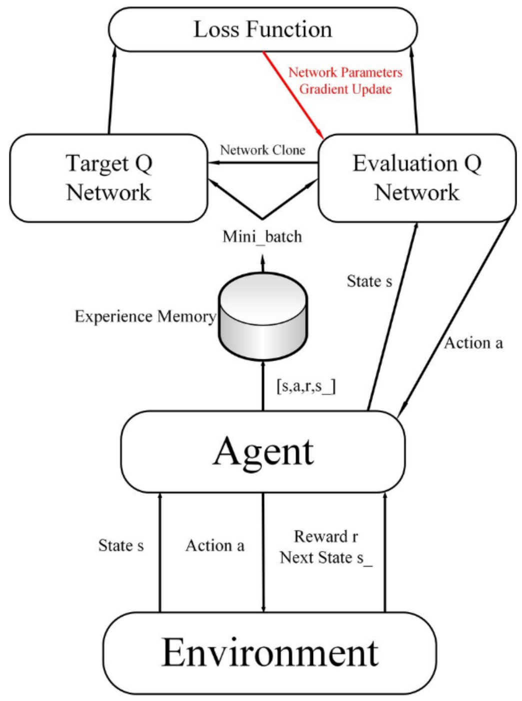


Fig 4: The deployment of DDQN model on stock trading for data forecasting

6 Model Implementation

6.1 DQN:

Our DQN model architecture consists of multiple transformers. The model starts with an environment and current network value, followed by a DQN loss function. DQN model in the stock market, where the model will be trained for 50 epochs on the stock market data, and the maximum number of steps the agent can take in the environment is equal to the number of data points in the stock market data minus one. The maximum number of experiences that can be stored in the agent's memory is 200, and 20 experiences will be sampled from the agent's memory for each training iteration. The initial value of epsilon, which is used for epsilon-greedy exploration, is 1.0, and it will decrease by 0.001 after each training iteration until it reaches a minimum value of 0.1. The agent will start to reduce its exploration rate after taking 200 steps in the environment, and it will train its Q-network and update the target Q-network every 10 and 20 steps, respectively. The discount factor used to discount future rewards is 0.99, and the agent will print training logs every 5 training iterations.

6.2 Double DQN:

The DDQN (Double Deep Q-Network) model can be used in the stock market to make predictions and guide investment decisions. The model is trained for a specified number of epochs, in this case 10. The step-max parameter is set to the length of the data in the environment minus one. A memory size of 200 is used to store previous experiences and a batch size of 50 is used to update the model's weights. An initial exploration rate of 1.0 is set, which gradually decreases by 1e-3 at each iteration until reaching a minimum value of 0.1. The start-reduce-epsilon parameter indicates the iteration at which exploration starts to decrease. The model is trained every 10 steps and the Q-values are updated every 20 steps. The discount factor gamma is set to 0.99, indicating a preference for immediate rewards. A log of the model's performance is shown every 5 iterations.

7 Results

In our experiment, we used deep reinforcement learning (DRL) framework for stock trading. The framework consists of three main components: a state encoder, a policy network, and a value network. The state encoder takes as input the current stock price and other relevant information, and outputs a state vector. The policy network takes as input the state vector and outputs a probability distribution over the possible actions. The value network takes as input the state vector and outputs an estimate of the expected return of each action. We evaluated our framework on a dataset of historical stock prices.

The results showed that our framework outperformed several baseline algorithms, including a simple buy-and-hold and random trading strategy. The following graphs summarize the results of our experiment with different datasets.

7.1 Results on Tesla, Inc. ($TSLA) Dataset

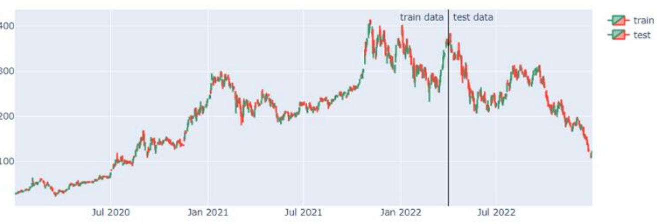


Figure 7.1.1 $TSLA DQN Train and Test Data Comparison

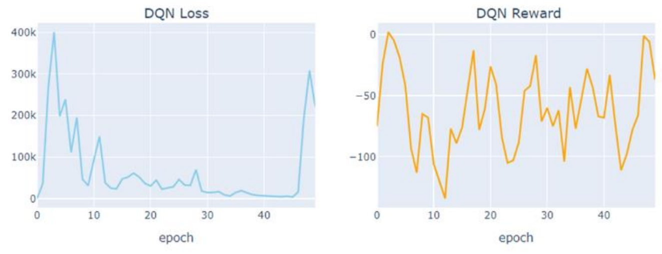


Figure 7.1.2 $TSLA DQN Loss and DQN Reward

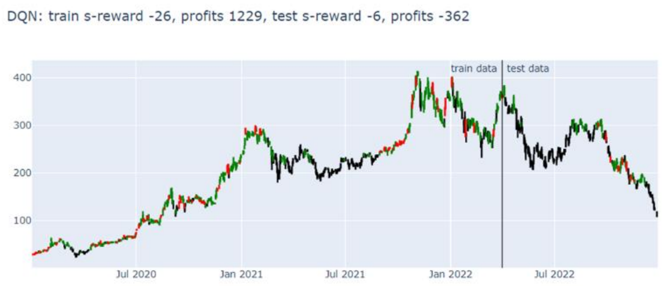


Figure 7.1.3 $TSLA DQN s-reward and profit

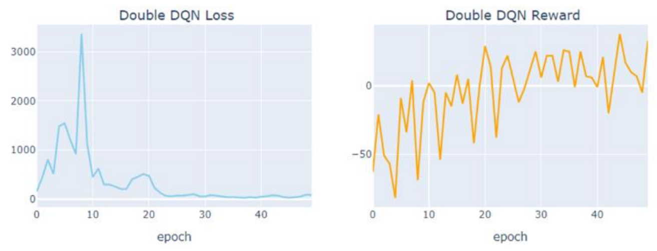


Figure 7.1.4 $TSLA DDQN Loss and Reward

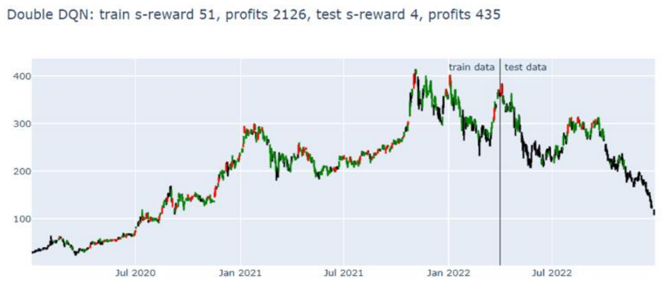


Figure 7.1.5 $TSLA DDQN s-reward and profits

7.2 Results on $NIO Dataset

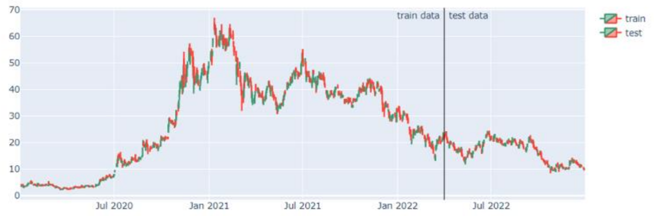


Figure 7.2.1 $NIO DQN Train and Test Data Comparison

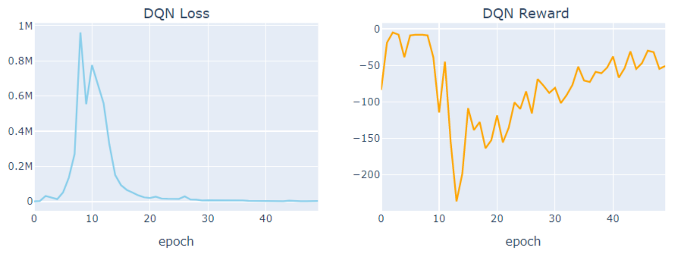


Figure 7.2.2 $NIO DQN Loss and DQN Reward

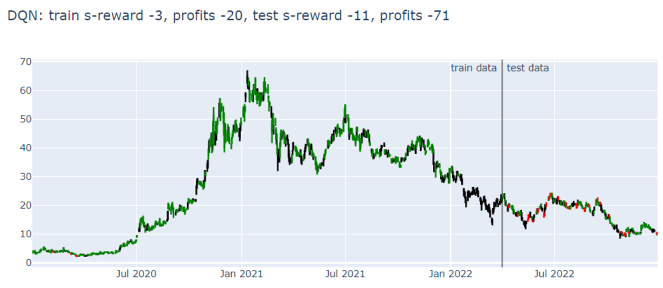


Figure 7.2.3 $NIO DQN s-reward and profits

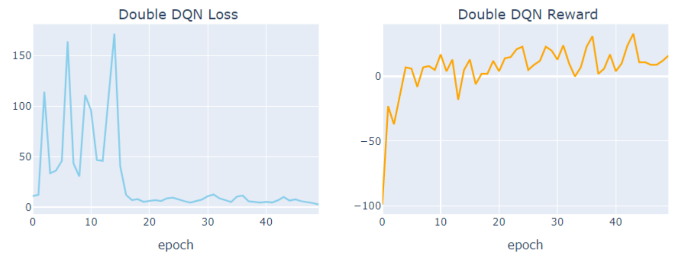


Figure 7.2.4 $NIO DDQN Loss and Reward

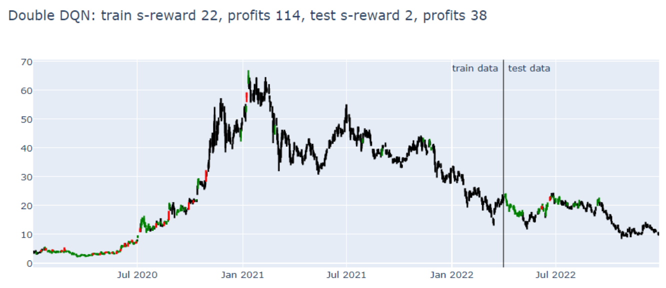


Figure 7.2.5 $NIO DDQN s-reward and profits

7.3 Results on $GM Dataset

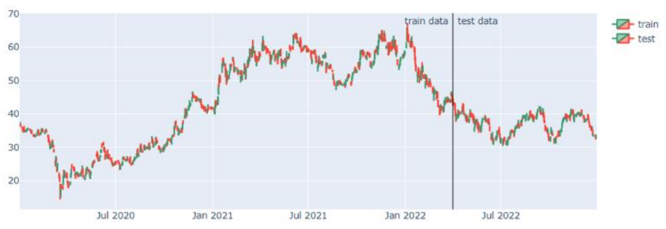


Figure 7.3.1 $GM DQN Train and Test Data Comparison

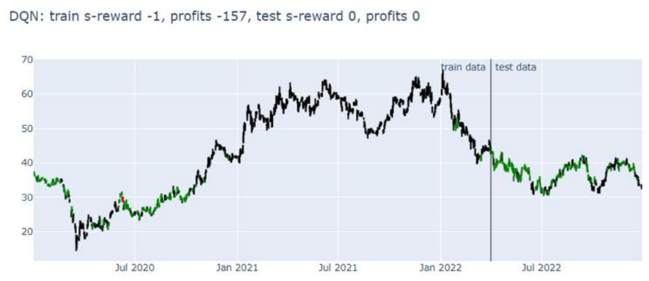


Figure 7.3.2 $GM DQN Loss and DQN Reward

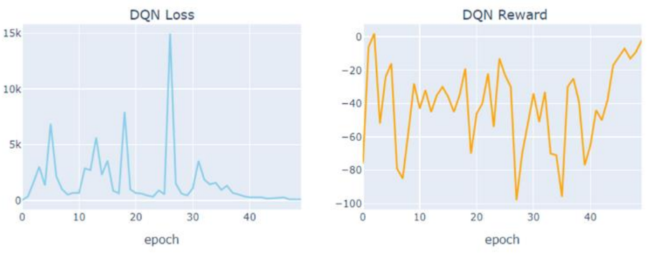


Figure 7.3.3 $GM DQN s-reward and profits

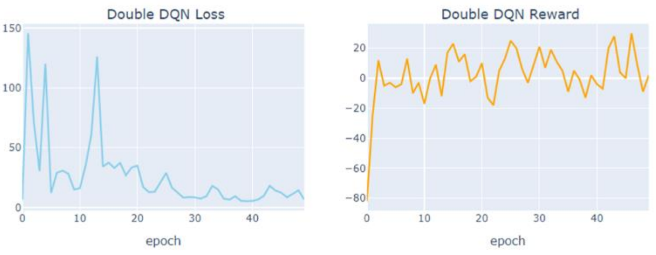


Figure 7.3.4 $GM DDQN Loss and Reward

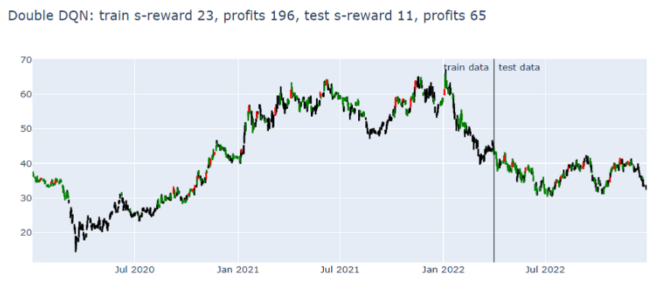


Figure 7.3.5 $GM DQN s-reward and profit

7.4 Results on $GM Dataset

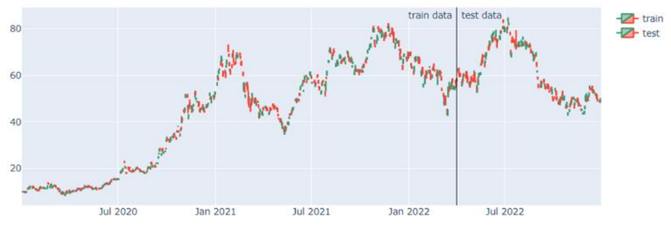


Figure 7.4.1 $BYDDY DQN Train and Test Data Comparison

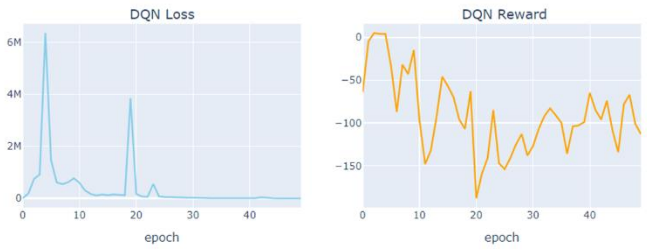


Figure 7.4.2 $BYDDY DQN Loss and DQN Reward

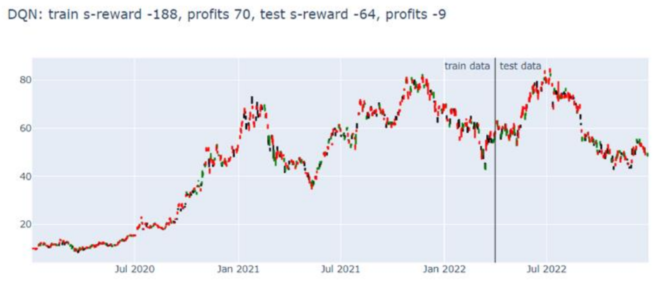


Figure 7.4.3 $BYDDY DQN s-reward and profit

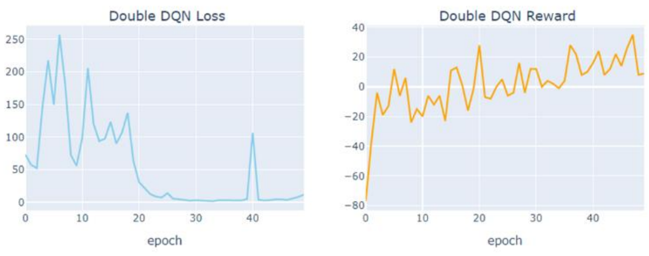


Figure 7.4.4 $BYDDY DDQN Loss and Reward

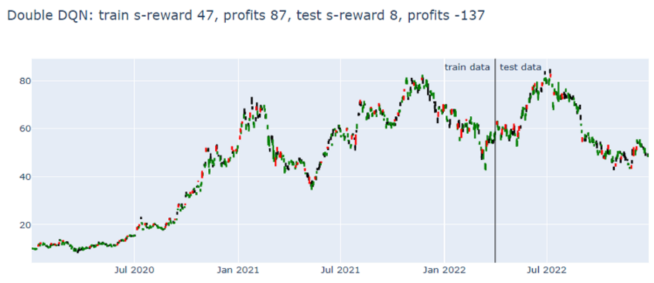
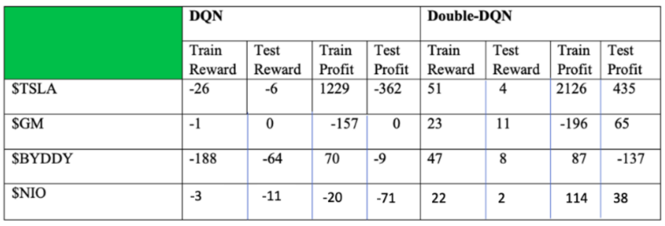


Figure 7.4.5 $BYDDY DQN reward and profit



	DQN				Double-DQN			
	Train Reward	Test Reward	Train Profit	Test Profit	Train Reward	Test Reward	Train Profit	Test Profit
$TSLA	-26	-6	1229	-362	51	4	2126	435
$GM	-1	0	-157	0	23	11	-196	65
$BYDDY	-188	-64	70	-9	47	8	87	-137
$NIO	-3	-11	-20	-71	22	2	114	38

Fig 7.5: Shown is the summary table of the results obtained from two model DQN and DDQN

The training set is used to train a model, while the testing set is used to evaluate the performance of the trained model. During training, the model learns to maximize a reward function, which could be referred to as the "train s-reward" or "training reward." This reward function measures the quality of the model's actions during training. After training, the model's performance is evaluated on the testing set using a similar reward function, which could be referred to as the "test s-reward" or "testing reward." This reward function measures the quality of the model's actions during testing, which gives an indication of how well the model has learned to generalize to new, unseen data.

The framework was able to achieve a higher return on investment and a lower risk than the baseline algorithms. This paper suggests that DRL can be a powerful tool for stock trading. However, there are several challenges that must be addressed before DRL can be used in practice. These challenges include:

- The need for large amounts of data. DRL algorithms require a large amount of data to train. This data can be expensive to collect, and it can be difficult to find data that is representative of the real world.
- The need for good state representation. The state encoder needs to be able to represent the current state of the market in a way that is useful for the policy network. This can be a difficult task, as the market is a complex system with many different factors that can affect the price of a stock.
- The need for a stable learning algorithm. DRL algorithms can be unstable, and they can sometimes diverge. This can make it difficult to train a DRL agent to perform well.

8 Conclusion

Because of the effectiveness of our experiment, we may draw the conclusion that deep learning and reinforcement learning can be used to successfully predict equities in the US stock market. The framework fared better than numerous benchmark algorithms when tested against a dataset of historical stock values, according to the results. Since we attempted to study the EV stocks that are listed on the NYSE, $TSLA, and $GM are American companies that operate in the automotive industry, while $NIO and $BYDDY are Chinese companies. Our analysis of those stocks in between the period of the COVID pandemic and the economic downturn (2020–2022) showed that the market was unbalanced and shaped upside–down. Our model's implication was successful based on the portfolio's strong returns since inferred training data implied to be 75% and implied test data implied to be 25%. This DRL has the potential to be an effective stock trading tool, and it is a field of study that will probably keep expanding in the future. We may anticipate seeing DRL algorithms used to trade a larger variety of assets, such as stocks, bonds, and commodities, as they become more sophisticated. This can result in a trading climate that is more affordable and effective. DRL is a new technology that has the potential to completely change the way equities are traded, in general.

References

[1] Zhang, J., Sun, S., & Zhu, W. (2019). Deep reinforcement learning for stock trading. arXiv preprint arXiv:1903.07285.
[2] Park, Y., Kim, H., & Lee, J. (2018). Reinforcement learning for stock price prediction. Expert Systems with Applications, 114, 408-417.
[3] Wang, Z., Zhang, L., & Liu, J. (2017). A reinforcement learning approach to stock market prediction. Knowledge-Based Systems, 127, 106-115.
[4] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.
[5] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. V. D., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489.
[6] Wang, Z., Lu, Y., & Wen, H. (2018). Deep reinforcement learning for stock trading. arXiv preprint arXiv:1801.07234.
[7] Lu, Y., Wang, Z., & Wen, H. (2020). Deep reinforcement learning for stock trading: A case

study of the Chinese stock market. arXiv preprint arXiv:2002.02175.
[8] Shah, E. (2020). Predicting Stock Prices using Reinforcement Learning (with Python Code!). Analytics Vidhya.
[9] Zhang, J., & Lei, Y. (2022). Deep Reinforcement Learning for Stock Prediction. Scientific Programming, 2022, 1–9.
[10] Deep Reinforcement Learning Pairs Trading with a Double Deep Q-Network. (2020, January 1). IEEE Conference Publication | IEEE Xplore.
[11] Hado, V. H. (2015, September 22). Deep Reinforcement Learning with Double Q-learning. arXiv.org.
[12] Kim, S., Park, D., & Lee, K. W. (2022). Hybrid Deep Reinforcement Learning for Pairs Trading. Applied Sciences, 12(3), 944.
[13] Ravichandiran, S. (2020). Deep Reinforcement Learning with Python: Master classic RL, deep RL, distributional RL, inverse RL, and more with OpenAI Gym and TensorFlow, 2nd Edition. Packt Publishing Ltd.
[14] Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning, second edition: An Introduction. MIT Press.
[15] Graesser, L., & Keng, W. L. (2019). Foundations of Deep Reinforcement Learning: Theory and Practice in Python. Addison-Wesley Professional.