DQN stock tradebot using Macro-econoimc Data

Bomin Zhang
Science Academy
Univeristy of Maryland, College Park
College Park, United States
bominz2@umd.edu

***Abstract*—This project explores the development of a reinforcement learning (RL) stock trading bot using a Deep Q-Network (DQN) framework. The bot takes as input various stock indicators, macroeconomic data from the Federal Reserve Economic Database (FRED) [1], and simulated account information. The DQN architecture leverages a policy network for action selection and a target network for stable learning, along with experience replay and an epsilon-greedy exploration strategy. Results indicate that while the DQN-based bot outperforms baseline strategies (e.g., buy-and-hold) during certain market conditions, but still it sometimes underperforms when market dynamics shift. Future improvements include expanding the action space (partial positions, short selling, multi-asset trading) and training of model that consider multiple stocks at the same time.**

Keywords—RL, DQN, macro-economic data, Algorithmic Tarding

I. Introduction

Algorithmic trading and quantitative investing have garnered significant attention in both academic and industry settings. The ever-growing availability of high-frequency market data, combined with advances in computing power, makes data-driven decision-making approaches particularly appealing. Reinforcement Learning (RL), which focuses on learning optimal actions through trial-and-error interactions with an environment, offers a natural fit for stock trading — where decisions about buying, selling, or holding stocks translate into sequential actions with long-term implications. In this project, we integrate macroeconomic indicators from FRED, stock technical indicators, and simulated account information to create an environment suitable for a Deep Q-Network (DQN) agent. The research question we address is: Can a DQN-based RL agent utilizing macrod economic data outperform naive strategies (e.g., buy-and-hold)?

II. Objectives

A. To incorporate both technical and fundamental (macroeconomic) data into the RL state representation.

B. To implement and train a DQN-based RL agent that learns optimal trading actions over a wide timeframe (2000 – 2020) and evaluate its performance (2020 – 2021, and 2020 – 2024).

C. To compare the RL agent’s performance to buy-and-hold strategies, as well as to previously tested RNN-based models. (the latter of which is not completed; due to lack of time for adaptation)

The **scope** of this study limited to single-stock trading decisions, and the agent either go all in or all out (doesn't consider partial positions). The significance lies in showcasing how reinforcement learning with market indicator & macroeconomic signals is capable of outperforming buy-and-hold strategies.

III. Literature Survey

I didn't peroform a literature survey on my own; Instead used a survey paper by Ali Alameer et al. (2022) [2] as a reference. From which I explored L. Chen & Q. Gao (2019) [3] *Application of Deep Reinforcement Learning on Automated Stock Trading* as a reference for the DQN algorithm and general consideration in implementation.

Previous RNN-based attempts: Prior attempts using RNN for price prediction underperformed, possibly due to a mismatch in time-series modeling vs. sequential decision making. An RL approach directly focuses on the reward structure of trading, which may provide better results. The codebase can be found in “/depricated/” folder of the project's codebase.

IV. Data Collection And Processing

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

A. Data Collection

1) Stock Candle Data

Minute-level stock data from Alpaca-trade API; eventually switched to daily data from Yahoo Finance (yfinance) [4], since the macroeconomic data is more suitable for intraday trading (as opposed to more frequent trading).

2) Stock Considered

Initially top 50 volume stocks from the S&P 500. Initially went with current top 50 volume stock from S&P500; Later realized taht this likely is not a good idea, since these stock doesn't represent the market as a whole very well, and

they likely have experience more growth in general than the market as a whole.Later switched to all stocks added in the S&P 500 prior to 2000 and not yet removed (*this left a set of 177 stocks*).

3) Macroeconomic Data from FRED

a) Federal Funds Effective Rate [5]

The Federal Funds Rate is a key interest rate that influences the cost of borrowing money in the United States. It is the rate at which banks lend to each other overnight to meet reserve requirements.

b) University of Michigan Consumer Sentiment [6]

This index measures consumer sentiment about the economy and personal finances.

c) Consumer Price Index (CPI) for All Urban Consumers [7]

This index tracks the average change in prices for a basket of goods and services purchased by urban consumers.

d) Retail Sales [8]

This measures the total sales of retail goods and services, including online sales.

e) Capacity Utilization: Total Index [9]

This index measures the percentage of total potential output that is being used, indicating how much of the economy's resources are being utilized.

More macroeconomic data remain to be added; This is probably not a good set, would require help from economist for advice on an influential set of macroeconomic data.

B. Data Preprocessing & Feature Engineering

1) Technical Indicators

Technical indicators including price trends, price channels, oscillators, stop-and-reverse signals, and candlestick patterns. The calculation of these indicators are done using the `stock-indicators` python package [10].

2) Data Splits

Training*: 2000-01 to 2019-12

Testing*: 2020-01 to 2021-01 (primary out-of-sample test); additional test 2020-01 to 2024-01.

3) Normalization

Applied feature scaling to technical and macroeconomic indicators. Most are normalized with respect to the close price of the stock, and others are normalized using mean and standard deviation according to what they model.

4) Feature Dimensionality

State vector includes technical indicators, macro variables, and account info (e.g., cash balance, current position). The current version has a state of 17(indicators)+15(macro) = 32 features.

DQN Implementation

I used a relatively naive implementation of DQN:

A. Policy/Target Network

a) a naive 3 layer fully connected NN used in target and policy network of DQN

b) Input layer: 32(#feature)+2(account information: relative cash storage, average position price)=34 input, 64 output.

c) Hidden layers: One fully connected layers with 64 units, ReLU activation.

d) Output layer: Taking 64 input, output 3 Q-values corresponding to the discrete actions (hold, buy, sell).

B. Deep Q Network

a) Experience Replay: Stores tuples (state, action, reward, next_state, done) in a replay buffer. Random sampling breaks correlation and improves data efficiency.

b) Target Network: Periodically updated to match weights of the policy network, stabilizing training.

c) Epsilon-Greedy Exploration: Epsilon starts at 1.0 and decays with rate 0.999.

d) Loss Function: Mean Squared Error (MSE) of Q-value estimates vs. target Q-values.

e) Optimizer: Adam, learning rate assumed to be 1e-3.

Simulation and Back Testing System

The back testing and simulation system comprises two key components: the Account class for portfolio management and the *TradingSimulator* class for trading execution. Together, these components facilitate the evaluation and training of reinforcement learning (RL) agents using historical market data in a controlled environment.

A. Portfolio Management System: Account Class

The Account class manages the portfolio's cash balance, positions, orders, and associated fees. It maintains an accurate representation of the agent's portfolio state and enables robust financial metric calculations.

Key Features:

a) Cash and Position Tracking

Maintains the cash balance and asset holdings, updating positions and average prices when trades occur.

b) Order Execution

Supports the execution of buy and sell orders, including verification of order validity and the deduction of transaction fees.

c) Fee Calculation

Implements realistic transaction costs, including:

i. SEC Fees (proportional to order value)

ii. TAF Fees (based on shares traded, subject to a cap).

d) Portfolio Metrics

Provides critical financial metrics, such as:

i. Total Portfolio Value: Sum of cash and the market value of held positions.

ii. Cash Ratio: Proportion of portfolio value held as cash.

iii. Position Weights: Contribution of individual assets to the total portfolio value.

The latter two are normalized and used as part of the input state for DQN network.

B. Trading Execution Environment: TradingSimulator Class

The TradingSimulator provides an environment for simulating the execution of trading actions within a backtesting framework. It manages order fulfillment, market price interactions, and reward calculations based on agent performance. Key features include:

a) Mock Trade Data Retrieval

Simulates stock price data one step ahead to evaluate the feasibility of order execution without exposing future information to the agent.

b) Action Execution

Implements the following discrete actions: Hold (no trade), Buy, and Sell.

c) Ensures order validity & filling order

Verify sufficient cash for buying or shares for selling and cancels unfilled orders. Determines whether an order is executed based on simulated next day price conditions:

i. Buy Orders: Filled if the current price is less than or equal to the order price.

ii. Sell Orders: Filled if the current price is greater than or equal to the order price.

d) Reward Calculation

Computes rewards based on

i. Changes in portfolio value: $\Delta V = V - V'$

ii. Penalty for liquidity issues (e.g., asset concentration): $L = \mathrm{std}(R)$

iii. Invalid or unfilled order penalty.

And the reward is

$$\alpha \times \Delta V - \beta \times L - \gamma$$

Where α, β, γ are respective multipliers that are hyper parameters.

C. Simulation Workflow

a) The RL agent selects an action (hold, buy, or sell) at each time step.

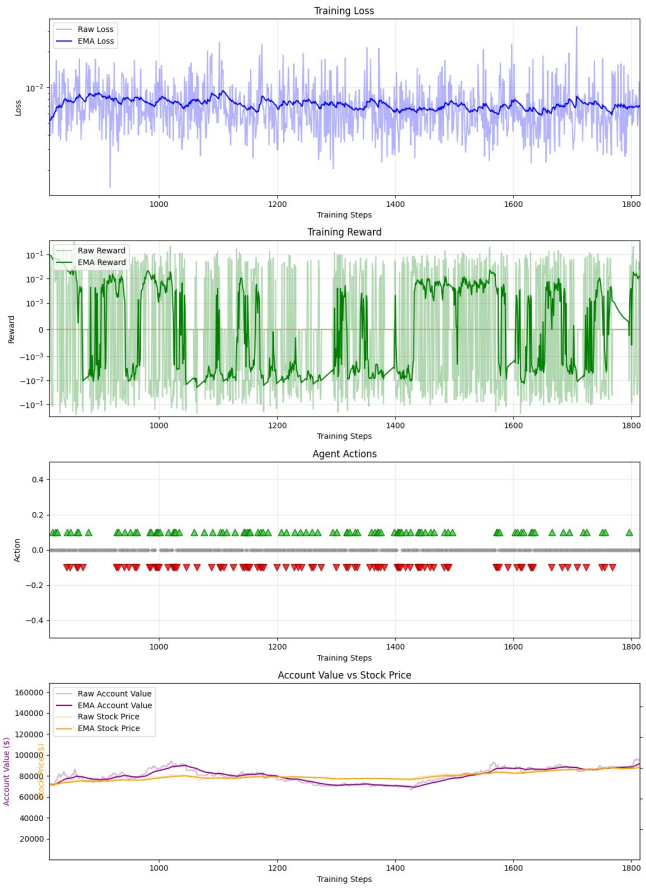
b) The simulator retrieves the relevant market price and executes the chosen action.

c) Order validity is verified, and filled/unfilled orders are processed.

d) The reward is calculated based on performance metrics and penalties.

e) The portfolio state is updated, and the agent receives feedback for learning.

The proposed backtesting and simulation system provides a robust and realistic environment for evaluating RL-based trading strategies. It incorporates historical market data, portfolio management, realistic transaction fees, and reward-based feedback mechanisms. This framework enables agents to learn effective trading policies through interaction, order execution, and reward optimization within a simulated market environment.



Example of training period

RESULT ANALYSIS

After training the model with feature for 177 stock data & 5 macroeconomic data between 2000~2020 for around 250 epochs, here is the test result:

A. Performance Metrics

a) Return

i. The percentage change in the account value.

b) Buy and Hold Return

Used as **baseline**; The percentage change in the account value if the agent had simply held the stock; In other words, the return of the stock.

c) Outperformance

Strategy Return – Buy and Hold Return

d) Relative Outperformance

Outperformance / Buy and Hold Return

e) Utilization Rate

Average percentage of the account's total value that is invested in the market.

f) Utilization-Adjusted Outperformance

Strategy Return - (Utilization Rate * Buy and Hold Return)

g) Utilization-Adjusted Relative Outperformance

Utilization-Adjusted Outperformance / Buy and Hold Return

h) Average Number of Trades

For analyzing trading frequency

B. Testing Result

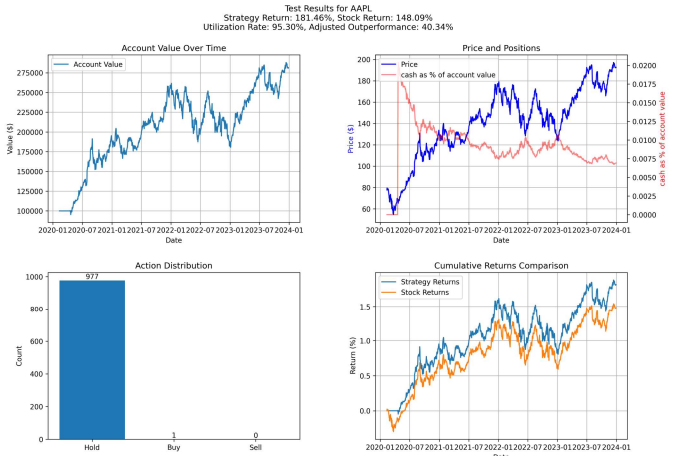
a) With 2020~2021 data:

i. Average Strategy Return: 12.82%
ii. Average Stock Return: 3.95%
iii. Average Strategy Outperformance: 8.87%
iv. Average Relative Outperformance: 162.25%
v. Fund Utilization Rate: 61.19%
vi. Utilization-Adjusted Outperformance: 11.38%
vii. Utilization-Adjusted Relative Outperformance: 410.83%
viii. Average Number of Trades: 5.9

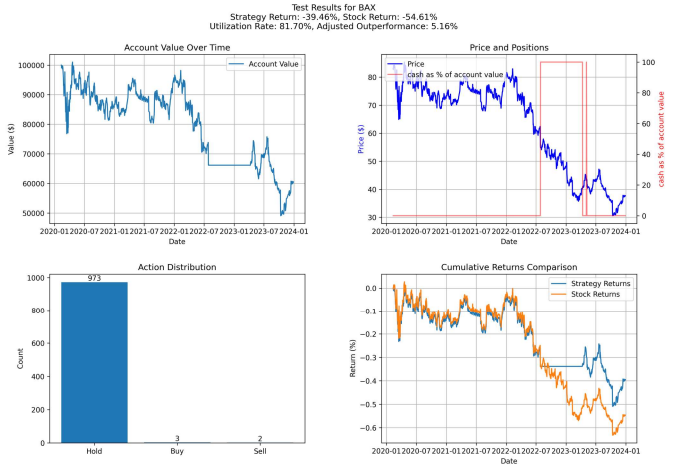
b) With 2020~2024 data:

i. Average Strategy Return: 41.86%
ii. Average Stock Return: 43.50%
iii. Average Strategy Outperformance: –1.64%
iv. Average Relative Outperformance: 223.89%
v. Fund Utilization Rate: 80.55%
vi. Utilization-Adjusted Outperformance: 8.67%
vii. Utilization-Adjusted Relative Outperformance: 272.34%
viii. Average Number of Trades**: 32.6

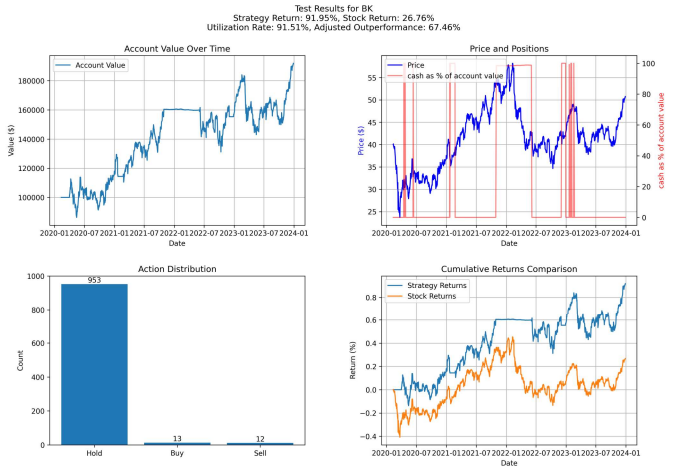
During the major downturn in early 2020, the agent tended to wait for market recovery signals before entering positions, effectively mitigating losses. But afterwards, the strategy slightly under performed the broader market.



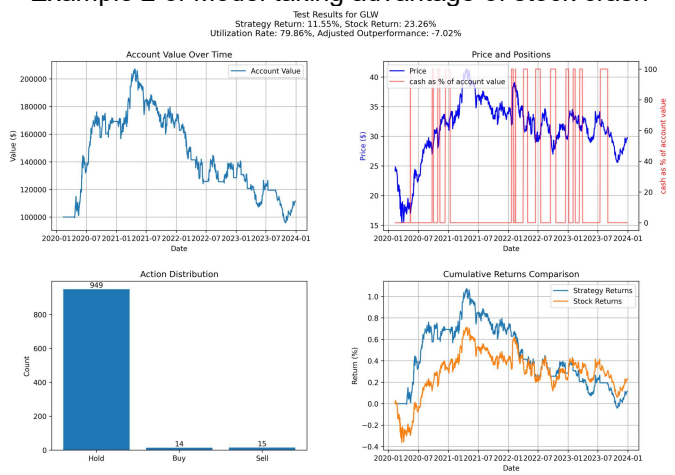
Example of the model avoiding initial stock crash



Example 1 of Model taking advantage of stock crash



Example 2 of Model taking advantage of stock crash



Example of model underperforming

Two plausible cause for this observation:

The model is over-fitting to the economic situation before 2020; (and is not able to properly understand the change in stock market during the AI boom); This can be addressed naively by updating the model to keep up with more recent training data.

The model is better at preventing loss than making profit (as its training data experiences both the dot-com bubble and 2008 financial crisis). It performs better during the 2020~2021 period where there is major stock market downturn, while underperforming the market during the AI boom.

CONCLUSIONS

A. Key Findings

a) This DQN-based RL agent integrating macroeconomic and technical indicators can outperform naive strategies under certain market conditions (e.g., 2020 downturn).
b) Performance can lag behind buy-and-hold when markets trend strongly upward (e.g., the AI-driven rally).

B. Limitations

a) Single-stock focus, ignoring portfolio diversification.
b) Fixed action space (full buy, full sell, or hold). Potential over-fitting to historical crises.

C. Future Work

a) Expand the action space to include partial buy/sell and short selling.
b) Incorporate RNN or LSTM modules for better time-series feature extraction.
c) Provide sector and industry context as input features.
d) Online retraining to adapt to new market conditions.
e) Multi-asset, portfolio-based reinforcement learning.

REFERENCES

[1] Federal Reserve Economic Data https://fred.stlouisfed.org/

[2] L. Chen and Q. Gao, "Application of deep reinforcement learning on automated stock trading," in 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS). IEEE, 2019, pp. 29–33.

[3] Alameer, A., Saleh, H., & Alshehri, K. (2022). *Reinforcement Learning in Quantitative Trading: A Survey.* TechRxiv. March 10, 2022.

[4] yfinance https://pypi.org/project/yfinance/

[5] Board of Governors of the Federal Reserve System (US), Federal Funds Effective Rate [FEDFUNDS], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/FEDFUNDS, December 15, 2024.

[6] University of Michigan, University of Michigan: Consumer Sentiment [UMCSENT], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/UMCSENT, December 15, 2024.

[7] U.S. Bureau of Labor Statistics, Consumer Price Index for All Urban Consumers: All Items in U.S. City Average [CPIAUCSL], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/CPIAUCSL, December 15, 2024.

[8] U.S. Census Bureau, Retail Sales: Retail Trade [MRTSSM44000USS], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/MRTSSM44000USS, December 15, 2024.

[9] Board of Governors of the Federal Reserve System (US), Capacity Utilization: Total Index [TCU], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/TCU, December 15, 2024.

[10] stock-indicators https://python.stockindicators.dev/guide