

Article

A Hybrid Framework for Multi-Stock Trading: Deep Q-Networks with Portfolio Optimization

Soroush Shahsafi  and Farnoosh Naderkhani * 

Concordia Institute for Information Systems Engineering (CIISE), Concordia University,
Montreal, QC H3G 1M8, Canada; soroush.shahsafi@mail.concordia.ca

* Correspondence: farnoosh.naderkhani@concordia.ca

Abstract

This paper presents a hybrid framework for multi-stock trading that combines the decision-making ability of Deep Q-Networks (DQN) with the allocation precision of portfolio optimization models. Realistic markets are noisy and non-stationary, and complex action spaces can hinder reinforcement learning (RL) performance. The DQN generates buy/sell signals based on market conditions. The framework passes buy-listed assets to an optimizer, which computes portfolio weights. Five allocation strategies are examined: naïve 1/N, Markowitz Mean–Variance, Global Minimum Variance, Risk Parity, and Sharpe Ratio Maximization. Empirical evaluations on emerging-market exchange-traded funds (ETFs), as well as additional tests on U.S. equities, show that even the baseline DQN outperforms traditional technical indicators. Furthermore, integrating any of the optimization approaches with DQN yields measurable improvements in return-risk performance metrics. Among the hybrid frameworks, DQN combined with Sharpe Ratio Maximization delivers the most consistent gains. The findings highlight the value of decomposing stock selection from capital allocation and demonstrate the effectiveness of the proposed DQN-optimization framework on our testbed.

Keywords: Deep Q-Networks (DQN); reinforcement learning; stock trading; portfolio optimization; sharpe ratio maximization; emerging market

1. Introduction

Portfolio management is an important part of financial management where the goal is to manage risk and return. The Markowitz Mean-Variance framework has proved that diversification reduces risk while maximizing this risk-return trade-off (Markowitz, 1952). This framework helped investors evaluate risk and return. Despite its appeal, classical portfolio theory often fails in real markets that require sequential decisions under uncertainty. This challenge becomes even greater when trying to learn which assets to trade in a multi-asset universe with non-stationary market dynamics.

To navigate this complexity, investors have historically relied on two major approaches for stock trading: fundamental analysis and technical analysis. Fundamental analysis is still the foundation of stock trading and involves an evaluation of equities based on their intrinsic value, taking into consideration such factors as financial statements, economic indicators, and market conditions. This method is based on the idea that stock prices will eventually align with the company's true value thus generating profit chances when prices are different from their intrinsic value. On the other hand, technical analysis uses charts and volume to predict short-term price moves. This approach is based on two assumptions:



Academic Editors: Wei Li and
Ninghui Liu

Received: 30 December 2025

Revised: 2 February 2026

Accepted: 4 February 2026

Published: 9 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article
distributed under the terms and

conditions of the [Creative Commons
Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

that market prices reflect all available information and that market patterns tend to repeat over time. Both approaches have limits: technical signals can mislead, and fundamentals often react too slowly for short horizons (B. Huang et al., 2019).

Recent machine learning (ML) methods process large datasets and uncover patterns that traditional methods miss. Building on the theme of machine learning applications, P. Li and Yang (2025) introduced a novel informed trading index for the Chinese A-share market using an ensemble neural network. Their index, derived from high-frequency trading data, strongly predicts future aggregate market returns. Recent deep learning models, including Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs), demonstrate greater effectiveness in financial context. For instance, CNNs can transform time series data and technical indicators into image-like structures for detecting patterns that are not obvious through analytical methods (Chandar, 2022), and LSTMs can better understand long-term dependency in sequential data compared with ARIMA, as evident in their application in predicting stock price directions (Ghosh et al., 2022).

Building on the potential of ML-based models enhanced by technical indicators, advanced approaches like Reinforcement Learning (RL) offer further innovation in financial markets. Reinforcement Learning has presented new opportunities in the financial markets, especially when integrated with the perceptual capabilities of Deep Learning. Deep Reinforcement Learning (DRL) is now an indispensable technique for creating dynamic and complex trading strategies that can respond to the ever-changing nature of financial markets (Y. Li et al., 2020; P. Liu et al., 2023). Unlike static models that may suffer from model bias due to their oversimplified assumptions, DRL models dynamically adapt by continuously learning from evolving market environments. This flexibility is important in the stock market due to its nonlinear and non-stationary data influenced by numerous external factors. However, RL agents often struggle with capital allocation across many assets and can generate noisy or concentrated portfolios; conversely, optimization techniques excel at allocation given a set of candidate assets but do not provide dynamic selection. This combination lets the agent select assets while the optimizer produces risk-aware allocations.

Despite significant advancements, stock trading still faces some challenges. ML-based models can be highly sensitive to input data, which may lead to overfitting and poor performance (Lv et al., 2019). The presence of noise in financial time series data can have an adverse impact on the performance of trading models. Since noise in the data may hide important patterns and lead to poor trading decisions, noise reduction is a necessary step in financial data analysis. Among the existing methods for noise reduction in financial data, the Wavelet Transform (WT) is one of the most widely used (Dastgerdi & Mercorelli, 2022; Lee et al., 2021). While the Fourier Transform can only provide frequency information, the WT provides both time and frequency information, which makes it more appropriate for the analysis of non-stationary data such as financial time series data. In WT, data is split into high-frequency (detail) and low-frequency (trend) components, which helps models to focus on the most informative part of the data. This is a relevant approach as it allows for capturing specific market conditions and improving decision-making in trading activities. It has been indicated that the incorporation of WT with DRL models can enhance the accuracy of market prediction and profitability, especially in volatile financial markets (Lee et al., 2021). There are other methods of noise elimination such as the Kalman Filter, which are also prevalent in financial modeling. Such methods enhance the accuracy and reliability of the financial models by filtering out noise while retaining the structure of the data (Dastgerdi & Mercorelli, 2022).

To the best of our knowledge, no previous studies have conducted a comparative analysis of portfolio optimization methods integrated with reinforcement learning algorithms such as Deep Q-Networks (DQN). This paper seeks to address this gap by combining

DQN with a range of portfolio optimization frameworks, such as the naïve 1/N portfolio, Markowitz Mean-Variance, Global Minimum Variance, Risk Parity, and Sharpe Ratio Maximization. Through a systematic comparison of these optimization methods in the context of reinforcement learning, this work contributes to the identification of the robustness of DQN in dynamic multi-stock trading scenarios. This paper shows that the hybrid DQN-optimization framework is capable of reacting to the actual conditions in real-time, thus achieving superior results in managing portfolios and trading.

The remainder of this paper is structured as follows: Section 2 reviews both traditional and modern stock trading approaches, applications of reinforcement learning, and challenges related to noisy data. In Section 3, we discuss the employed techniques, including reinforcement learning fundamentals, the Deep Q-Learning algorithm, the denoising approach, and portfolio optimization strategies. Section 4 presents the experimental setup and results, covering data sources, evaluation criteria, benchmark strategies, and an in-depth analysis of the findings. Finally, Section 7 provides conclusions and outlines potential directions for future research.

2. Literature Review

Portfolio optimization remains one of the primary areas of concern in quantitative finance, where the goal of investors is to optimize multiple objectives: maximize returns while diversifying different types of risks. During the past two decades, multi-objective optimization has been recognized as an efficient approach in many areas, especially when dealing with complex decision environments. A comprehensive survey of non-dominated sorting-based methods highlights how Pareto-optimal reasoning has become a foundational component of modern multi-objective algorithms, enabling systematic exploration of trade-offs across competing objectives and supporting applications in diverse domains, including finance and engineering (Seada & Deb, 2018). Within financial applications, multi-objective optimization has been extensively used to extend classical portfolio theory beyond single-objective formulations. In this context, mean-variance optimization provides a canonical risk–return baseline and is closely linked to empirical tests of portfolio efficiency and asset-pricing validation. A widely used formal procedure is the Gibbons–Ross–Shanken (GRS) test, which evaluates whether a given portfolio is mean-variance efficient. Recent work by (Agrawal, 2023) recasts the GRS statistic in a trigonometric/geometric framework, providing additional interpretation and implementation guidance. Beyond the mean-variance setting, uncertainty-aware portfolio models have been proposed to address the limitations of probabilistic assumptions in real markets. In particular, a mean-entropy portfolio selection framework formulates expected return and risk as competing objectives under uncertainty theory, generating a Pareto front of nondominated portfolios rather than a single solution (Majumder et al., 2018). Several studies further emphasize the importance of robust allocation under uncertainty or imprecision. Interval-based multi-objective formulations extend classical optimization by allowing objective values to be represented as ranges, capturing ambiguity in financial inputs and market conditions. An extended nondominated sorting genetic algorithm (ENSGA-II) has been proposed to handle such interval-valued objectives through interval ranking and interval-based crowding distance, showing improved robustness compared to standard NSGA-II (Bhunia et al., 2018).

Although these optimization frameworks provide principled ways to control trade-offs and model uncertainty, they are typically formulated as static allocation problems and do not naturally address the sequential decision-making inherent in multi-stock trading environments. The stock market remains a challenging place for traders and investors because of the nature of the market and the large volumes of information. Financial trading in the past was a process where human beings were the main decision-makers and relied on

technical indicators, fundamental analysis, and qualitative data. However, the introduction of ML and AI technologies has significantly changed this scenario. Recent studies leverage machine learning to extract predictive signals in China's financial markets. [L. Wu et al. \(2023\)](#) used deep learning to analyze analysts' textual industry reports, constructing an industry sentiment metric that generates abnormal returns via long-short portfolios.

Recent work applies RL to trading and shows that agents can adapt strategies to changing markets. This is particularly true since RL models can adapt to changing market conditions by leveraging real-time data. [Y. Li et al. \(2022\)](#) is among the studies that showcased the effectiveness of RL in trading. In this study, RL was integrated with a CNN to analyze historical price data to select stocks. The CNN contributed to transforming the raw price data into appropriate features like moving averages, price momentum, and volatility, which the RL agent employed to make trading decisions. Using S&P 500 stocks, the proposed model outperformed other conventional strategies over 21 years, with a higher return on investment and a higher Sharpe ratio. This paper was able to prove that RL models could be effective in the long run if they are used with large datasets and diversified stocks. In [\(P. Liu et al., 2023\)](#), the author included attention mechanisms into the RL framework where the agent can gauge the relevance of features like high, low, and close prices. In natural language processing, attention mechanisms enable models to focus more on relevant information and less on the insignificant points. This was particularly beneficial in volatile domains like trading in cryptocurrencies, as the model had to adapt to the changes in price very frequently. This study showed that this attention-driven RL model was more effective than traditional CNNs in trading by capturing market sentiment and volatility changes and demonstrated the significance of feature prioritization in RL-based trading techniques.

Recently, there is an interesting development is the Multi-type Data Fusion Framework based on DRL (MSF-DRL), which allows to consider a variety of financial data for a better understanding of market trends and thus, a better trading strategy [\(P. Liu et al., 2023\)](#). Also, [Biondo et al. \(2024\)](#) developed a multi-type data fusion framework. This has incorporated stock price data, technical indicators, and candlestick charts into a single framework of RL. The spatial and temporal aspects of the model are captured by CNNs and LSTM networks, respectively, thereby reducing the impact of noise and improving the performance of their framework.

State and action spaces in RL models are among the factors that determine both the speed and the capability of learning. Continuous action spaces, in which the agent chooses the exact number of stocks to purchase or sell, are more flexible but require more computation time. This poses a significant challenge to training since agents have to consider a much larger space of actions. [Betancourt and Chen \(2021\)](#); [Y. Li et al. \(2020\)](#) discuss the advantages of having a discrete action space to facilitate learning. Their models were very effective and could perform well, even when interfering factors, such as stock market fluctuations, were present.

Researchers often improve rewards by adding risk terms, such as the Sharpe ratio. A reward function based on the Sharpe ratio is introduced in [\(Z. Liu, 2023\)](#) that combines profit and risk into the learning process. The Sharpe ratio measures performance adjusted for risk, which discourages high variability. Moreover, having this metric in the process, the agent can learn about returns but also about risks, which in turn helps to create balanced portfolios.

Financial markets are considered noisy environments where price movements depend on a large number of factors that are difficult to predict. In trading, this kind of noise is one of the biggest challenges when applying RL since it hampers the overall decision-making process. Noise can hide signals and push RL agent toward overfitting or unstable decisions or make it overly sensitive to random fluctuations in prices. [Liang et al. \(2023\)](#) address

the challenges of modeling high-frequency financial data by proposing a hybrid wavelet-deep learning framework (DB-BLSTM). Their model, designed to handle nonlinearity and complex periodicity in intraday stock index futures, outperforms traditional linear models and standalone deep-learning approaches. Shavandi and Khedmati (2022) proposed a hierarchical multi-agent system where agents were divided based on the time horizon they were accorded (short-term, medium-term, long-term agents). This structure is based on the fractal market hypothesis, which helps agents to get information from different periods and thus differentiate between noise and signal. The study conducted using Deep Q-Network (DQN) revealed that this type of multi-agent system was much more efficient than the conventional single-agent based systems, as it could manage noisy financial data more efficiently, especially in foreign exchange markets such as EUR/USD. Lee et al. (2021) solved the noise problem by using wavelet transformation (WT) for pre-processing of financial data. WT dissects stock prices into their frequency elements, which in effect filters out noise, or high-frequency data from the actual price movement, or low-frequency data. The pre-processed data is then passed to the LSTM network, which enhances the overall performance of the trading system. The empirical analysis of the study thus reveals that the application of denoising techniques greatly improves RL-based trading, particularly under conditions of high uncertainty. X. Wu et al. (2020); Yang (2023) also address noise by implementing hierarchical RL systems, which break down the complexity of portfolio management and reduce sensitivity to market fluctuations through policy layering. Although this approach is effective in managing noise, it also introduces complexity in training, which can become prohibitive as the number of assets increases.

Recent literature has explored both end-to-end deep reinforcement learning (DRL) allocators and two-stage (hybrid) pipelines that separate asset selection from portfolio weighting. Early DRL studies typically formulate allocation as the agent's direct action, either by discretizing portfolio weights or by using continuous-action frameworks (e.g., actor-critic variants) that jointly learn selection and weighting over the full asset universe (Gao et al., 2020; Jiang et al., 2017). In contrast, a growing body of two-stage work first pre-filters or ranks assets using a predictive model and then applies a classical optimizer to determine portfolio weights; these studies often report improvements over pure-optimization baselines and highlight the practical benefits of decoupling selection from allocation (S. Huang et al., 2024; Wang et al., 2024). More recent comparative investigations evaluate multiple RL architectures and hybrid variants, showing that design choices involve important trade-offs in sample efficiency, scalability, and robustness (Cui et al., 2024). Our contribution differs from this prior work in two key respects: (i) we use DQN strictly as a discrete selection agent rather than as a continuous weight predictor; and (ii) we couple that selection stage with a controlled comparison of five allocation strategies within a single pipeline to isolate the incremental impact of the allocation rule, with robustness tests across emerging-market ETFs and U.S. equities.

While there has been tremendous progress in employing RL in financial trading systems, the current literature still lacks sufficiently precise solutions for the multi-stock trading problem when it comes to scalability and computational complexity. Most models suffer from high computational costs, especially when dealing with continuous action spaces or complex portfolios covering multiple assets. This paper addresses that gap by proposing a hybrid framework that explicitly decouples discrete buy/sell selection (performed by a DQN) from continuous weight determination. By reducing the RL action space and leveraging well-understood optimization methods for risk control and regularization, the approach improves interpretability and computational tractability. Crucially, we provide a systematic comparison of five allocation strategies integrated within the same

DQN-selection pipeline, empirically evaluating their return-risk tradeoffs and robustness across asset classes.

3. Material and Methods

3.1. Reinforcement Learning

In RL, an agent learns sequential decisions by interacting with an environment. The agent learns a policy that maximizes long-run reward. In contrast to supervised learning where models are trained on labelled data, RL employs continuous experimentation with the environment and is encouraged by feedback in the form of recompense. The agent and its environment can be presented as a Markov Decision Process (MDP) where the next states are decided based only on the current state and the action taken that makes the process Markovian. The agent aims at getting the highest possible return R which is the cumulative of the discounted reward in an episode (Equation (1)).

$$R(\tau) = \sum_{t=0}^T \gamma^t r_t, \quad (1)$$

where γ is the discount factor and r_t is the reward received at time step t .

The agent learns a policy $\pi(a|s)$, that defines the likelihood of choosing action a when in state s . The value function $V^\pi(s)$ represents the expected return starting from state s and following the policy π . Similarly, the Q-function $Q^\pi(s, a)$ provides the expected return for performing action a in state s and then following the policy π .

One of the challenges in RL is the balance between exploration and exploitation. Exploration is the process of taking new actions to find out the consequence, while exploitation uses known actions that yield high rewards. Exploration leads to the creation of new solutions by the agent, while exploitation allows for the immediate optimization of results already available to the agent. One of the common exploration strategies is called the ϵ -greedy policy: with probability ϵ the agent chooses a random action while with probability $1 - \epsilon$ it takes the action with the highest estimated value.

3.2. Deep Q-Networks (DQN)

The Deep Q-Network or DQN is a value-based Q-learning algorithm, which uses a deep neural network as a function approximator for estimating the optimal action-value function $Q^*(s, a)$. In classical Q-learning, the Q-values are learned iteratively using the Bellman Equation (2) to update the Q-table.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right) \quad (2)$$

where α is the learning rate, r_t is the reward at the time step t , and γ is the discount factor. However, the Q-learning algorithm does not easily generalize to the high dimensionality of the state space and therefore is not as useful for problems such as multi-stock trading.

In DQN, the Q-function is approximated by a deep neural network which takes the current state as the input and then produces the Q-values of all the actions. During training, the agent updates its Q-values based on the Bellman equation, but with a key difference: instead of using the Q-values from the next state, the agent utilizes a different target network which is a copy of the original Q-network but with a delay to ensure that the training does not diverge when updating the Q-values.

We train the DQN by minimizing the MSE between predicted and target Q-values (Equation (3)).

$$L(\theta) = \mathbb{E}_{(s,a,r,s')} \left[(y_t - Q(s_t, a_t; \theta))^2 \right] \quad (3)$$

where $y_t = r_t + \gamma \max_{a'} Q'(s_{t+1}, a'; \theta^-)$ is the target Q-value computed using the target network with parameters θ^- , and θ are the parameters of the current Q-network. This loss is minimized by the agent where the error is backpropagated through the neural network and the weights are updated through gradient descent. We use Adam optimizer for its effectiveness in working with sparse rewards and MSE as the loss function.

The agent randomly selects experiences from the replay buffer for updating the Q-network. It also avoids learning based on consecutive experiences, which helps break the correlation between consecutive observations and smooth the learning process. The decision of which option to choose between exploration and exploitation is done using an ϵ -greedy policy. The agent chooses a random action with a certain probability called ϵ to ensure that the agent also has the capability to explore the state action space.

The replay buffer is a crucial aspect of DQN responsible for holding the experience tuples of (s, a, r, s', d) where s represents the current state, a is the chosen action, r is the reward obtained, s' is the subsequent state, and d is a binary variable indicating whether an episode is done. A replay buffer is used to make an agent focus on some specific transitions that occur rarely but are critical to learning. It also works to decorrelate the experiences due to random shuffling and taking random samples of the experiences which in turn makes the training process more stable. In the replay function of our implementation, the target Q-values are calculated based on the Bellman equation, where the target is computed in Equation (4).

$$\text{Target} = r_t + (1 - d_t) \cdot \gamma \cdot \max_{a'} Q(s_{t+1}, a') \quad (4)$$

These target Q-values are derived from the target network while the Q-values which are predicted are derived from the main Q-network. The Q-values are updated only for the action taken by the agent in the current state while the Q-values of other actions remain the same. This helps the network to direct the learning updates to the specific actions that were taken instead of updating all the Q-values. The agent's performance is measured in the sum of the reward collected over time and the learning process lasts for a set number of episodes. The weights in the model are saved at intervals, and when the agent is trained, it is ready to decide in a test scenario.

3.2.1. State Space

In the context of the reinforcement learning framework for stock trading, the choice of state space is one of the most important steps for the agent to make. The state space is used by the Deep Q-Network (DQN) and has a direct effect on the agent's perception and interaction with the market. In this work, we make improvements to the state space to overcome issues that are specific to noisy financial data and the large dimensionality of multi-stock price series. Daily closing prices, in particular, contain a lot of noise because of daily market activity, changes in the macroeconomic environment, and other unpredictable events. Using raw stock prices as the state space for the agent introduces two primary challenges:

- **Noisy Data:** Sometimes, it may be hard for the agent to differentiate between actual market trends and noise especially when price movements are characterized by minor fluctuations and randomness.
- **High Dimensionality:** When dealing with portfolios that contain multiple stocks, the direct use of raw prices for each stock implies a large and very unstructured state space that hampers the learning and slows it down to converge.

Under these conditions, it is necessary to preprocess the stock price data to minimize noise and simplify the structure of the state space.

To extract useful information from the historical stock prices, we use an attention-based neural network model. The attention mechanism is central to filtering out important movements in price from unimportant movements in price. Attention mechanisms have shown great potential in extracting meaningful features from financial time series data, as demonstrated by (Y. Li et al., 2022), where an attention-based transformer network is used to capture temporal dependencies and improve stock movement predictions. Similar approaches, such as Temporal Pattern Attention (TPA) applied to multivariate time series data in (Wei et al., 2020), highlight the importance of attention mechanisms in capturing complex relationships in stock price movements. The attention-based model is intended to work on windows of past stock prices. The input is a set of prices for a given time interval called the window size, and the output is a forecasted price that captures the key patterns within the sequence. The model receives input sequences of historical prices. We define a window size n , which determines how many consecutive days of stock prices are fed into the model. The most important component of the model is the Attention Layer which enables the network to pay attention to the most important price points in the input sequence. Instead of treating each day in the window equally, the attention mechanism provides weights to each price depending on its contribution towards the predicted result. The attention mechanism calculates a contextual vector based on all the time steps in the input sequence. This contextual vector focuses on the specific stock prices that are likely to be influential in the next movements while at the same time deemphasizing others that are less influential. Mathematically, given an input sequence X_t , the attention layer computes an attention score a_i for each time step i , and the final context vector c_t is obtained by Equation (5).

$$c_t = \sum_{i=1}^n a_i P_i \quad (5)$$

where P_i is the price at time step i in the input window, and a_i is the attention weight for that time step.

The output of the attention layer, that is the context vector, is fed to a Flatten Layer to make it more suitable for the final prediction. The flatten layer eliminates additional dimensions and gets the output ready for the next step. A Dense Layer with only one neuron is used as the output layer of the model. This dense layer yields a scalar prediction which is understood as the predicted price of the next step. The last layer is fully connected, so all the elements of the input vector are used for a final prediction, which allows the network to find subtle patterns in the price data. To train the model, several epochs are required, the batch size is adjusted to get the optimal convergence. During this training process, the attention mechanism assists in determining the weights that should be assigned to different time steps to help the model identify the important price changes while disregarding the noise. To determine the state space, we calculate the trend between the subsequent predictions from the attention-based model while having the refined price predictions. The trend is the percent change between two successive predictions. The state space at time t is a vector that includes the trend values for each stock in the portfolio. For a portfolio of m stocks, the state vector is defined based on the Equation (6):

$$s_t = [\text{trend}_1, \text{trend}_2, \dots, \text{trend}_m] \quad (6)$$

where trend_i represents the trend between the predicted prices at times $t - 1$ and t for stock i . This trend-based state captures the directional movement of stock prices, providing the agent with a clearer representation of market trends without being overwhelmed by noisy data or high-dimensional inputs.

3.2.2. Action Space

The action space refers to the set of all the choices available to the agent at any given time step. In this framework, the action space is discrete, that is, each action will be a decision about which stocks to purchase or sell. When designing the action space, one of the foremost decisions is whether to use discrete or continuous action space, which has its pros and cons. In a discrete action space, the agent is limited to selecting the specific predetermined actions including “buy” or “sell” for each of the stock. This simplifies the decision-making process but limits the discretion of the agent regarding the distribution of portfolio investment. For instance, the agent might choose to either purchase or sell a particular stock yet it will not actively determine the quantity of shares. This structure is computationally efficient and assists the agent in navigating the action space more efficiently. In a continuous action space, the agent not only decides what action to take (buy or sell) but also how much of it to undertake (number of shares to buy or sell). While this gives much more control over the construction of a portfolio, it makes the problem much more complex. The agent has to search a much larger, continuous space of actions than in discrete state space settings, which also slows the learning and increases computational complexity. Due to the complexity of the computational aspects and the scope of this framework, a discrete action space is chosen. Nevertheless, once the decision of which stocks to purchase has been made, an optimization procedure is called to decide the amount of money to be invested in the selected stocks. This hybrid approach simplifies the action space while ensuring that the stock allocation problem is solved optimally.

The framework uses RL for asset selection and optimization for allocation. The agent works in a small action space where it provides high-level decisions on whether to buy or sell particular stocks. When multiple buy signals occur, the optimizer allocates funds across the selected assets. This approach reduces the dimensionality of the action space to just which stocks the agent should trade, rather than how many shares exactly. After the selection of the buy list, the system uses one of the portfolio optimization techniques to determine the appropriate weights to assign to the stocks. We delegate allocation to the optimizer, which keeps the agent’s action space discrete and training efficient. This structure of separation of learning and policy allows for effective management of the portfolio and risk-adjusted returns without putting the constant burden of action space exploration on the agent.

3.3. Proposed Hybrid Framework

In the present study, we do not adapt the objective to the market regime. Instead, we apply the same portfolio optimization method across all market conditions. In other words, after the agent selects assets, we solve the same optimization problem with no regime-based adjustment applied to the entire dataset. Figure 1 shows the complete workflow of our hybrid RL–optimization framework. In this framework, the DQN agent will look at the market state and send a discrete buy or sell signal on assets. The optimizer uses the agent’s signals as constraints: buy assets can receive weights, and sell assets receive zero. In other words, the DQN is only concerned with the high-level decision of the form of which assets, whereas the optimizer deals with the continuous decision of how much to allocate. In our implementation the allocation module recomputes portfolio weights at every decision step (daily in our experiments) when the DQN indicates buy candidates. Also, the computed weights are then converted to executed order sizes within the environment. It should be mentioned that for simplicity, transaction costs are set to zero in all experiments.

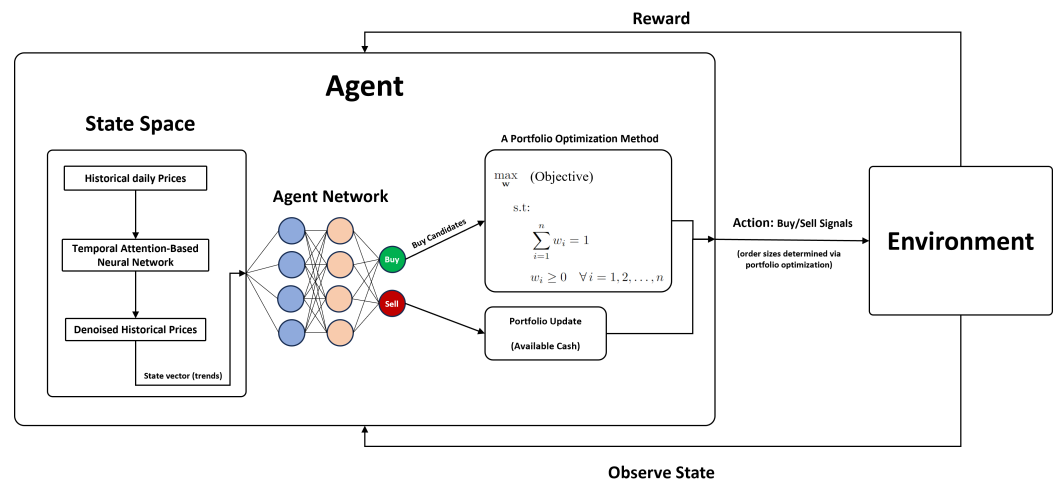


Figure 1. Workflow diagram of the proposed hybrid Deep Q-Network (DQN)-optimization framework.

Below, we provide a detailed overview of the five portfolio optimization methods implemented in this framework: The naïve $1/N$, Markowitz Mean-Variance, Global Minimum Variance, Sharpe Ratio Maximization, and Risk Parity.

The naïve $1/N$ equal-weight:

The naïve $1/N$ (equal-weight) is the simplest portfolio allocation strategy. Given N available assets, one simply assigns each asset a weight of $w_i = 1/N$. In this strategy, there is no need to estimate the expected returns, variances, or covariances. DeMiguel et al. (2009) showed that in many out-of-sample tests, the naïve $1/N$ strategy often outperformed more complicated optimizers that suffer from estimation error. While $1/N$ serving as a valuable benchmark, it is often enhanced with other techniques to better balance the trade-off between simplicity and economic efficiency.

Markowitz Mean-Variance Optimization:

The Markowitz mean-variance portfolio model, introduced by Harry Markowitz in 1952. It aims to make a trade-off between expected return and risk, where risk is measured by the variance (or standard deviation) of portfolio returns (Markowitz & Todd, 2000). Mathematically, it can be framed as a quadratic programming problem (Equation (7)).

$$\begin{aligned} \min_w \quad & \left(\frac{\gamma}{2} w^\top \Sigma w - w^\top \mu \right) \\ \text{s.t:} \quad & \sum_{i=1}^n w_i = 1 \\ & w_i \geq 0 \quad \forall i = 1, 2, \dots, n \end{aligned} \quad (7)$$

where $\gamma > 0$ is the investor's risk aversion parameter. When γ is small, the investor prioritizes higher expected return relative to variance. Conversely, a larger γ means that the investor emphasizes minimizing variance, resulting in a more conservative portfolio allocation. In this paper, we assume a moderate risk-aversion behaviour ($\gamma = 1$) for simplicity. Also, w is the vector of portfolio weights and Σ is the covariance matrix of asset returns.

Global Minimum Variance:

The Global Minimum Variance (GMV) portfolio is a specific optimal portfolio that has the lowest variance among all the portfolios on the efficient frontier. This portfolio aligns

with the preferences of an investor who is completely risk-averse and wants to minimize variance irrespective of the expected return. The importance of the GMV portfolio in financial applications was motivated by (Merton, 1980), who mentioned that variances and covariances of asset returns are much more accurately estimated than the estimation of means. The objective and constraints of this model are formulated in Equation (8).

$$\begin{aligned} \min_w \quad & w^\top \Sigma w \\ \text{s.t:} \quad & \\ & \sum_{i=1}^n w_i = 1 \\ & w_i \geq 0 \quad \forall i = 1, 2, \dots, n \end{aligned} \quad (8)$$

Sharpe Ratio Maximization:

The Sharpe Ratio, introduced by William Sharpe, is one of the most popular methods of measuring the risk-adjusted return of a portfolio. It refers to the measure of the extra return (over a risk-free rate) per unit of total risk, or standard deviation. Sharpe Ratio Maximization is based on achieving the maximum of this risk-adjusted return (Sharpe, 1998). The Sharpe Ratio SR is expressed in Equation (9):

$$SR = \frac{w^\top \mu - r_f}{\sqrt{w^\top \Sigma w}} \quad (9)$$

where:

- w is the portfolio weight vector.
- μ is the expected return vector.
- r_f is the risk-free rate.
- Σ is the covariance matrix of asset returns.

It should be mentioned that we utilized the yield on 3-month U.S. Treasury bills, obtained from the U.S. Department of the Treasury¹ as a proxy for the risk-free rate. We mapped these yields to our trading calendar and converted them to daily rates to ensure they are at the same frequency as our equity return data. The objective function and constraints of this model are illustrated in Equation (10):

$$\begin{aligned} \max_w \quad & \frac{w^\top \mu - r_f}{\sqrt{w^\top \Sigma w}} \\ \text{s.t:} \quad & \\ & \sum_{i=1}^n w_i = 1 \\ & w_i \geq 0 \quad \forall i = 1, 2, \dots, n \end{aligned} \quad (10)$$

Risk Parity:

The Risk Parity model intends to balance risks on every asset in the portfolio. In contrast to the concept of mean-variance optimization pioneered by Markowitz, Risk Parity seeks to make each asset contribute an equal portion of risk irrespective of potential returns (Roncalli, 2013). The risk contribution of each asset i to the total portfolio risk is defined in Equation (11):

$$RC_i = w_i \cdot (\Sigma w)_i \quad (11)$$

where $(\Sigma w)_i$ is the marginal risk contribution of asset i . The Risk Parity objective aims at producing an equal contribution of risk across the different assets and this can be expressed as minimizing the variance of these contributions (Equation (12)).

$$\begin{aligned} \min_w \quad & \sum_{i=1}^n \left(RC_i - \frac{\sigma_P}{n} \right)^2 \\ \text{s.t:} \quad & \\ & \sum_{i=1}^n w_i = 1 \\ & w_i \geq 0 \quad \forall i = 1, 2, \dots, n \end{aligned} \quad (12)$$

where σ_P is the total portfolio risk (standard deviation), and n is the number of assets.

3.4. Reward Function

The reward function offers information to the agent about the efficiency of its actions towards the end, which in the case of trading is normally the maximization of profits. The agent gets a numerical reward signal from the environment each time it takes an action which quantifies the immediate positive or negative value of the action. In the domain of multi-stock trading, this reward must be designed to reflect the change in the agent's portfolio. The reward function in the stock trading environment is usually associated with the financial outcomes of the trading actions. In this research, we define the reward R_t at time t as the change in the total portfolio value between two consecutive time steps. Formally, the reward at time step t can be expressed in Equation (13).

$$R_t = V_t - V_{t-1} \quad (13)$$

where V_t and V_{t-1} represent the portfolio values at time steps t and $t - 1$, respectively. The portfolio value V_t is computed as the sum of the cash in hand and the market value of all stocks held by the agent at time t (Equation (14)).

$$V_t = \sum_{i=1}^N S_i P_i + C \quad (14)$$

where:

- N is the total number of stocks in the portfolio,
- S_i represents the quantity of stock i held by the agent,
- P_i is the current price of stock i ,
- C is the available cash in hand.

4. Results and Experiments

4.1. Dataset

The dataset used in this study comprises three highly liquid, regionally diversified emerging-market ETFs:

- AAXJ (iShares MSCI All Country Asia ex Japan ETF) Captures broad Asia ex-Japan equity exposure, including China, India, and Southeast Asia.
- TUR (iShares MSCI Turkey ETF) Provides targeted access to Turkish large- and mid-cap equities, reflecting a emerging crossover market at the Europe–Asia nexus.
- EZA (iShares MSCI South Africa ETF) Offers focused coverage of South African equities, an important emerging-market economy.

These ETFs were selected to collectively span three major EM regions, deliver pronounced cross-region and sectoral diversification (technology and consumer in AAXJ; financials and industrials in TUR; materials and energy in EZA), and maintain high average daily trading volumes, ensuring data reliability. Limiting the study to three assets preserves computational tractability by constraining the dimensionality of the DQN action space.

To evaluate the liquidity of our selected ETFs, we benchmarked them against location-based peer sets of comparable regional and single-country ETFs. Using daily Yahoo Finance data over 2015–2024, we computed Amihud illiquidity for each ETF (lower values indicate higher liquidity) and ranked each focal ETF within its peer group. The Amihud illiquidity measure is the average of the daily ratio of absolute return to dollar trading volume over a chosen period (Amihud, 2002).

AAXJ was compared to FXI, EWY, MCHI, INDA, EWT, ASHR, EWH, EPP, EWM, EWS, THD, EPHE, AIA, EEMA, GMF, and IDX. It has mean Amihud illiquidity 1.890×10^{-10} and ranks 7/17 (62.5th percentile; better). EZA was compared to KSA, FM, EIS, EGPT, QAT, AFK, and UAE. It has mean Amihud illiquidity 1.269×10^{-9} and ranks 1/8. TUR was compared to KSA, FM, GREK, EGPT, QAT, AFK, and UAE. It has mean Amihud illiquidity 2.147×10^{-9} and ranks 1/8.

Daily adjusted closing prices were obtained from Yahoo Finance² for the period 1 January 2015 to 31 December 2024, and divided into training dataset (January 2015 to December 2022), covering various market regimes, such as pre- and post-pandemic, and a test dataset (January 2023 to December 2024) for the most recent market conditions in the post-pandemic era. Figure 2 illustrates the price movements for AAXJ, TUR, and EZA over the dataset's duration.

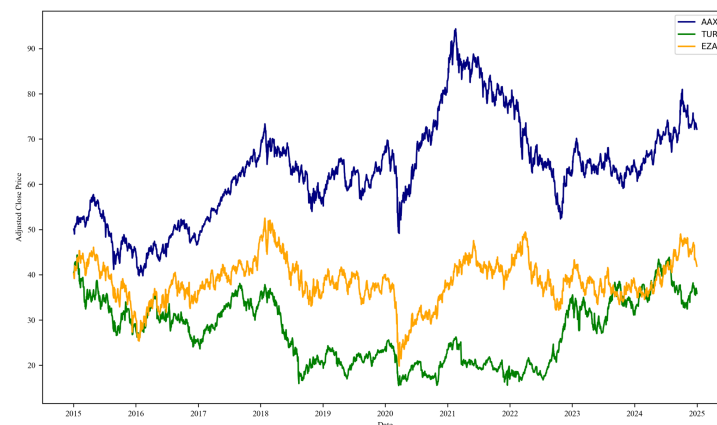


Figure 2. Historical adjusted close prices of the dataset including train and test periods (2015–2024).

4.2. Experimental Setup

All code was implemented in Python 3.10.18 (PyTorch environment) and trained on CPU. Our Q-network is a two-hidden-layer fully connected network with 16 neurons in each hidden layer, ReLU activations, and a final linear output layer with dimensions equal to the action space. As training proceeds, the exploration rate ϵ is reduced progressively. In our setup, the decay is regulated by the hyperparameter $\epsilon_{\text{decay}} = 0.9995$ and $\epsilon_{\text{min}} = 0.01$ so that the agent continues to explore the environment in the later phases of training. Moreover, the size of the replay buffer is 20,000 and training mini-batches are drawn randomly from the buffer with a mini-batch size of 128.

As shown in Table 1, we tune DQN hyperparameters using a sequential one-factor-at-a-time procedure. In other words, at each step, we changed a single hyperparameter over the candidate values while holding the remaining hyperparameters fixed. To mitigate overfitting, the training window is split chronologically into a train-inner subperiod and

a validation subperiod. For each candidate value, the agent is trained for 2000 episodes on train-inner and evaluated on the validation subperiod; the value that maximizes terminal portfolio value on validation is selected and carried forward to subsequent tuning steps. After tuning, all hyperparameters are fixed and all reported performance results are computed on the held-out test period only.

Table 1. Hyperparameter settings.

Hyperparameter	Tested Range	Selected Value
Number of hidden layers	{1, 2, 3, 4, 5}	2
Hidden layer size	{8, 16, 32, 64}	16
Replay-buffer size	{10,000, 20,000, 50,000, 100,000}	20,000
Batch size	{32, 64, 128, 256}	128
Discount factor γ	{0.90, 0.95, 0.99}	0.95
ϵ -greedy decay rate	{0.99, 0.995, 0.999, 0.9995}	0.9995

For instance, increasing the number of hidden layers or the width of each layer gives the network greater capacity to model complex, nonlinear relationships in the data but also increases the risk of overfitting. Regarding batch size, larger batches produce smoother, lower-variance gradient estimates and therefore faster convergence in the initial stages. Yet if the batch is too large relative to buffer diversity, the network may converge to a suboptimal policy. Setting γ closer to one encourages the agent to optimize for long-term gain, improving cumulative return, but can slow down convergence. Finally, a faster ϵ -greedy decay rate reduces exploration too quickly, causing the agent to get stuck in suboptimal policies, whereas a very slow decay rate prolongs exploration and delays exploitation of good strategies.

4.3. Comparison of DQN Against Technical-Indicator Benchmarks

The primary objective of this study is to compare the simple multi-stock trading strategy of Deep Q-Networks (DQN) with four of the most common technical indicators: Moving Average, Moving Average Convergence Divergence (MACD), Bollinger Bands, and Relative Strength Index (RSI). For the case of the simple DQN model, the agent employs a sequential allocation strategy (i.e., buying one-by-one each asset flagged by buy signal) when more than one buy signal is generated in a day. Also, it should be mentioned that each technical indicator is applied to the same assets to generate buy-and-sell signals according to its specific indicator logic. Also, each technical indicator uses the same sequential allocation strategy as the basic DQN model. To avoid in-sample overfitting when tuning technical indicators, we use a time-ordered nested split within the training period. Specifically, we split the full dataset into a training/validation segment and a held-out test segment. Within the training/validation segment, we further split into an inner-training window and a validation window. For each candidate hyperparameter configuration in the grid, indicator signals are computed using only historical information, shifted by one day to prevent look-ahead bias, and evaluated only on the validation window. The best-performing configuration is then locked and evaluated once on the held-out test segment.

To ensure a fair comparison between the basic DQN strategy and technical-indicator benchmarks, all methods are evaluated under identical execution and portfolio-construction rules. Each strategy operates long-only and without leverage, and is fully invested whenever at least one buy signal is present. When multiple assets are simultaneously flagged as buy, all methods apply the same sequential allocation mechanism to deploy capital. Rebalancing is signal-triggered for all strategies, i.e., trades occur only when the strategy's buy/sell signals change. Consequently, the basic DQN can rebalance more frequently

than rule-based indicators because it generates more frequent signal changes; however, this difference reflects the strategies' signal-generation behavior rather than differences in constraints or allocation mechanics. Under these shared rules, performance differences in the baseline comparison are therefore attributable to the quality of the generated trading signals.

4.3.1. Moving Average (MA)

The Moving Average is one of the simplest methods for identifying price direction and is designed to smooth noisy price data. It computes the volume-weighted average price of a stock over a specified period to reduce fluctuations caused by short-term volatility. The Moving Average strategy commonly uses two moving averages—a short-term and a long-term average—whose crossover points determine buy or sell signals. A buy signal is generated when the short-term moving average rises above the long-term average, known as the golden cross, indicating that the current price is increasing relative to the historical mean and suggesting a bullish trend. Conversely, when the short-term moving average falls below the long-term moving average, a death cross occurs, generating a sell signal that reflects bearish market conditions.

The strength of the Moving Average strategy lies in its ability to capture long-term market trends that may be obscured by daily fluctuations. However, it tends to produce delayed signals when sudden market movements occur, which can reduce responsiveness.

To identify the optimal parameters for this indicator, we conducted a grid search over the training period. Specifically, we tested all combinations of short-term windows from 1 to 19 days and long-term windows up to 29 days. For each pair, we performed a rolling-window backtest to generate crossover-based buy and sell signals and computed the resulting terminal portfolio value. The configuration that delivered the highest investment value on the training set was selected for out-of-sample evaluation. Applying this strategy to the test dataset resulted in a final portfolio value of \$19,642.

4.3.2. Bollinger Bands

Bollinger Bands are a volatility-based indicator that consists of three lines: a middle band with two side bands at two standard deviations above and below the middle band. Bollinger Bands are constructed based on the belief that, on average, prices will return to the midline after some time and that the distance between the two outer bands represents market activity. A buy signal is generated when the price is at or intersects the lower Bollinger Band. This logic is derived from the assumption that prices tend to move within a channel, and anything below the lower band is likely to bounce back. A sell signal is generated when the price reaches or crosses the upper Bollinger Band, implying that the stock is overbought, its price has risen sharply, and it may be due for a correction. Bollinger Bands are most effective in recognizing periods of wide volatility ranges.

All parameters of this indicator are optimized entirely on the training dataset via a grid search: we swept the rolling-window length from 5 up to 40 trading days and varied the band-width multiplier between $0.5\times$ and $4.0\times$ (in increments of 0.1). For each candidate setting, we constructed the upper and lower bands, generated crossover-based entry and exit signals when the price pierced the bands, and recorded the resulting terminal investment value to identify the optimal combination. When implemented on the test data, the strategy yielded a portfolio value of \$13,829.

4.3.3. Relative Strength Index (RSI)

The relative strength index is a momentum oscillator that compares the magnitude of recent gains to recent losses over a specified look-back period. It produces a value within the range of 0 to 100. When the RSI rises above a predefined threshold, traders

see it as a sign of overbought conditions (sell signal). Conversely, when it falls below a predefined threshold, they interpret it as oversold conditions (buy signal). Over the training window, we tuned the RSI strategy via an in-sample grid search: we varied the look-back period from 5 to 50 days and tested buy and sell thresholds from 25 to 34 and 65 to 74 respectively. For each candidate triplet, we computed the RSI series, generated rolling-window entry/exit signals, and simulated end-of-period investment value. The parameter set that delivered the highest final investment value was selected as the optimal configuration. In our simulation, the RSI-based strategy generated final portfolio values of \$15,486 at the end of the test dataset.

4.3.4. Moving Average Convergence Divergence (MACD)

The MACD is a more versatile momentum-based indicator that measures the difference between two exponential moving averages; one calculated over a shorter lookback period and the other over a longer lookback period. It is designed to provide both the trend and momentum in one single unit, which makes it more important in the technical analysis. MACD consists of three components:

- MACD line which is the difference between the shorter EMA and the longer EMA.
- Signal line which is a smoothed EMA of the MACD line, calculated over a shorter smoothing period.
- Histogram representing the difference between the MACD line and the signal line, shown as bars.

The MACD line above the zero line often depicts a bullish trend while a MACD line below the zero line shows a bearish trend. A buy signal occurs when MACD line goes above the signal line suggesting that the stock is in an upward momentum. A sell signal is derived when the MACD line goes below the signal line suggesting a weakening in the bullish pressure or the presence of bearish pressure. We applied the same in-sample grid search to MACD as to MA. During the training period, we exhaustively tested all combinations of (i) short EMA spans from 5 to 20 days, (ii) long EMA spans from 10 to 40 days, and (iii) signal-line smoothing spans from 5 to 19 days. The combination that yielded the highest terminal investment value was selected as the optimal configuration. The MACD strategy yielded a portfolio value of \$15,368 at the end of the test dataset. Figure 3 presents the test-period price history for each asset, together with buy and sell signals produced by the MACD indicator.

Also, Figure 4 presents the final portfolio value at the end of the test dataset for all four trading indicators and the basic DQN model. As it is shown, our basic DQN framework outperforms other technical indicators in generating buy and sell signals demonstrating the advantages of an agent that learns directly from market dynamics rather than relying on predefined rules.

4.4. Statistical Significance Between the Basic DQN and Technical Indicators

To address concerns about potential overfitting in benchmark strategies and to quantify the uncertainty of out-of-sample results, we performed statistical comparisons between the basic DQN strategy and each technical-indicator benchmark on the held-out test period. We used the daily portfolio value time series produced by each strategy and compute daily percentage returns. Specifically, we constructed 95% confidence intervals (CIs) for cumulative return, annualized volatility, Sharpe ratio, and maximum drawdown using a moving-block bootstrap of daily returns. In addition, we tested whether the mean daily return of the DQN strategy exceeds each benchmark using a Newey–West HAC *t*-test.

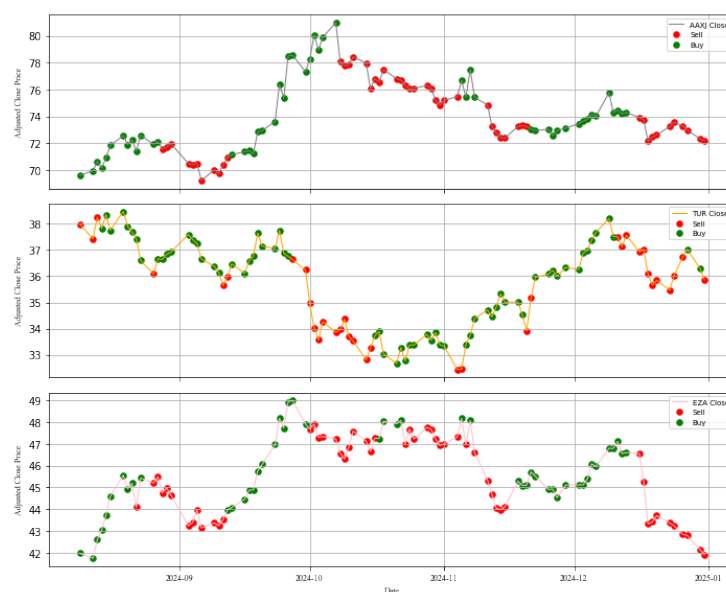


Figure 3. Test-period closing prices of AAXJ, TUR and EZA, annotated with buy (green) and sell (red) signals generated by the MACD indicator.

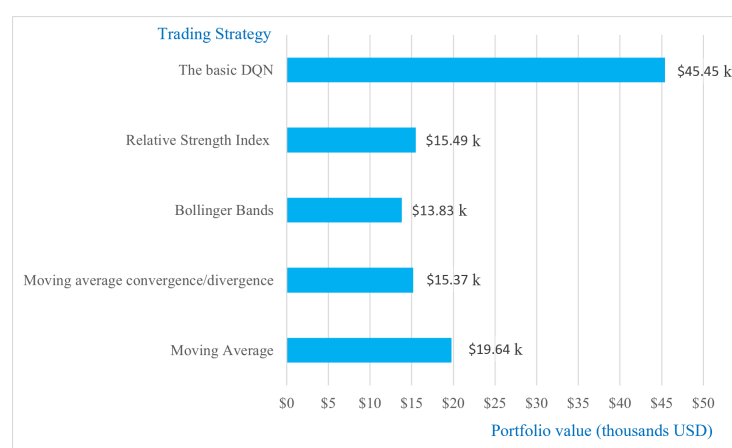


Figure 4. Comparison of the final portfolio value at the end of the test dataset for all trading indicators as benchmark strategies and the basic DQN model.

Across the held-out test period, the basic DQN delivers the strongest performance among all benchmark strategies. Table 2 summarizes the test-period metrics. The DQN achieves a cumulative return of 3.545 with annualized volatility 0.253, Sharpe ratio 3.131, and maximum drawdown 11.3%. In contrast, the technical-indicator strategies exhibit substantially lower cumulative returns (0.383–0.964) and Sharpe ratios (0.789–1.691), with larger drawdowns in most cases (11.5–19.9%).

Table 2. Test-period point estimates for the basic DQN and technical-indicator benchmarks.

Strategy	Cumulative Return	Volatility	Sharpe Ratio	Maximum Drawdown (%)
Basic DQN	3.545	0.253	3.131	11.3
MA	0.964	0.214	1.691	18.1
RSI	0.549	0.151	1.530	11.5
Bollinger Bands	0.383	0.244	0.789	19.9
MACD	0.537	0.238	1.024	17.3

To provide additional information about trading frequency, we also report the annualized turnover rate of each benchmark strategy (technical indicators) as well as the basic DQN over the held-out test period. Here, annualized turnover is a standard proxy for how actively a strategy trades, interpreted as the approximate fraction of the portfolio's value that is traded over a year. The estimated annualized turnover is 11.02% for RSI, 16.58% for Bollinger Bands, 49.90% for MACD, 63.57% for Moving Average, and 79.97% for the basic DQN. In addition, we report the number of executed buy/sell signals over the same test window: RSI executes 21 signals (11 buys, 10 sells), Bollinger Bands 33 (17 buys, 16 sells), MACD 141 (80 buys, 61 sells), Moving Average 152 (84 buys, 68 sells), and the basic DQN 172 (87 buys, 85 sells). Taken together, both turnover and executed-signal counts indicate that the basic DQN trades more actively than the rule-based indicator benchmarks under the shared sequential allocation rule.

To assess whether these differences reflect robust out-of-sample improvement rather than sampling noise, Table 3 reports block-bootstrap confidence intervals for the performance differences along with Newey–West HAC tests on the mean daily return difference series. This estimator produces standard errors that are robust to both autocorrelation and time-varying volatility. For every benchmark, the 95% bootstrap confidence intervals for both cumulative return and Sharpe ratio differences are strictly positive, indicating consistent outperformance by DQN. The Newey–West tests also reject equality of mean daily returns in favor of DQN for all four benchmarks (all $p < 0.004$). Because we perform four comparisons (DQN vs. four benchmarks), we control the family-wise error rate using the Holm–Bonferroni step-down procedure. Based on the results, all Holm-adjusted p -values remain < 0.005 , so the significance of the improvements is robust to multiple testing.

Table 3. Out-of-sample statistical comparison between the basic DQN and technical-indicator benchmarks on the test period.

Benchmark	ΔCR (95% CI)	ΔSR (95% CI)	NW t	p	Holm p
MA	2.769 [0.699, 6.211]	1.496 [0.311, 2.692]	2.882	0.00396	0.00396
RSI	3.185 [1.005, 6.901]	1.657 [0.264, 3.098]	3.500	0.000465	0.00140
BB	3.350 [1.244, 6.847]	2.398 [1.035, 3.683]	3.462	0.000536	0.00107
MACD	3.197 [1.241, 6.527]	2.163 [0.923, 3.445]	4.006	6.18×10^{-5}	2.47×10^{-4}

These results show that the basic DQN strategy delivers statistically significant improvements in mean daily returns relative to each technical-indicator benchmark. While DQN exhibits higher annualized volatility than MA and RSI in this test period (with positive bootstrap CIs for the volatility differences), the Sharpe ratio improvements remain statistically robust, indicating that the return gains are not solely explained by increased risk.

4.5. Hybrid DQN with Portfolio Optimization

In a follow-up experiment, we tested the effect of combining the DQN model with various portfolio optimization techniques. More specifically, instead of sequential allocation strategy (i.e., buying one-by-one each asset flagged by buy signal), we computed an optimal weight vector across all assets flagged by the buy signal using a portfolio optimization strategy. The assessment of the DQN model is done by several significant financial performance indicators that measure the profitability of the model and its ability to manage risk. These metrics are annual and represent important tools to compare the relative efficiency of various portfolio optimization techniques. Below, we discuss each metric with accompanying formulas and provide a comparison using results from test dataset (2023–2024).

Annual Cumulative Return measures the total net value of the portfolio for the financial year, expressed as a percentage of the original portfolio size (Equation (15)).

$$\text{Cumulative Return} = \left(\frac{P_{\text{end}}}{P_{\text{start}}} \right) - 1 \quad (15)$$

where P_{end} is the portfolio value at the end of the period, and P_{start} is the portfolio value at the beginning of the period.

Variance is a relative measure of variation of returns from the portfolio mean, showing the extent of variability. It is the standard deviation of returns, annualized through Equation (16).

$$\text{Volatility} = \sigma_{\text{daily}} \times \sqrt{252} \quad (16)$$

where σ_{daily} is the standard deviation of daily returns, and 252 is the average number of trading days in a year.

The Sharpe Ratio measures how much excess return the portfolio produces relative to risk (standard deviation), calculated as shown in Equation (17).

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma} \quad (17)$$

where R_p is the portfolio return, R_f is the risk-free rate, and σ_p is the portfolio's volatility.³

Maximum drawdown is the largest percentage decline in portfolio value from a prior peak to a subsequent trough during a given period. It measures downside risk and is computed in Equation (18).

$$\text{Maximum Drawdown} = \frac{\text{Peak Value} - \text{Trough Value}}{\text{Peak Value}} \quad (18)$$

Based on Table 4, combining a portfolio optimization strategy with the basic DQN model results in higher returns across all configurations. For instance, the basic DQN achieves a cumulative return of 3.545, and adding a naïve 1/ N (equal-weight) allocation increases it to 3.890, representing a 9.7% improvement. The hybrid DQN-Sharpe Ratio Maximization approach delivers the strongest gain, raising the cumulative return to 4.459, which is a 25.8% enhancement over the basic DQN while maintaining the same level of risk.

Volatility remains almost unchanged for both the basic DQN and the hybrid DQN-Sharpe Ratio Maximization approach; however, the hybrid DQN-Naïve 1/ N and DQN-Global Minimum Variance methods exhibit slightly lower volatility. When examining risk-adjusted performance and drawdowns, the hybrid DQN-Sharpe Ratio Maximization approach again stands out as the clear leader.

Table 4. Performance metrics for the emerging-market ETF test dataset (2023–2024).

Hybrid DQN + Portfolio Optimization	Cumulative Return	Volatility	Sharpe Ratio	Maximum Drawdown (%)
Basic DQN	3.545	0.2535	3.1315	11.31
DQN + Naïve 1/ N (Equal-Weight)	3.890	0.2429	3.4129	11.08
DQN + Risk Parity	3.687	0.2522	3.2090	10.07
DQN + Global Minimum Variance	3.811	0.2499	3.2893	11.30
DQN + Markowitz Mean Variance	3.848	0.2537	3.2596	10.08
DQN + Sharpe Ratio Maximization	4.459	0.2545	3.4864	9.71

To compare the results of the five portfolio optimization methods, the portfolio value at the end of the testing period has been calculated for each strategy. This metric presents an

easy approach to measuring each strategy’s general growth and profitability when applied to the test data set. As summarized in Table 5 the Sharpe Ratio Maximization method achieves the highest portfolio value at \$54,589, followed by the Naïve 1/*N* (Equal-Weight) approach at \$48,900 and the Markowitz Mean Variance method at \$48,480.

Table 5. Final portfolio values for combinations of DQN and portfolio optimization methods on the emerging-market ETF test dataset (2023–2024).

Hybrid DQN + Portfolio Optimization	Final Portfolio Value (Average)	% vs. Basic DQN
Basic DQN	\$45,448	—
DQN + Naïve 1/ <i>N</i> (Equal-Weight)	\$48,900	+7.6%
DQN + Risk Parity	\$46,871	+3.1%
DQN + Global Minimum Variance	\$48,111	+5.8%
DQN + Markowitz Mean Variance	\$48,480	+6.7%
DQN + Sharpe Ratio Maximization	\$54,589	+20.1%

Besides the average of final portfolio values, Figure 5 shows a histogram of the final portfolio values in all the episodes for the DQN–Sharpe Ratio Maximization framework. Relative to this distribution, it offers a better insight into how the portfolio value changes in multiple runs. The lowest value of the portfolio in this distribution is \$43,693, and the maximum value is \$63,452, which shows that the proposed framework consistently achieves high performance in the different episodes, even in the worst case. This result underscores the advantage of explicitly optimizing the trade-off between return and volatility by using excess returns over the risk-free rate, as the Sharpe Ratio Maximization framework does. While Markowitz’s mean–variance approach also balances these two dimensions, it requires careful calibration of the risk-aversion parameter γ to achieve comparable consistency. In contrast, the Sharpe Ratio Maximization objective inherently finds the portfolio with the best excess-return-per-unit-risk, delivering robust performance without the need for parameter fine-tuning.

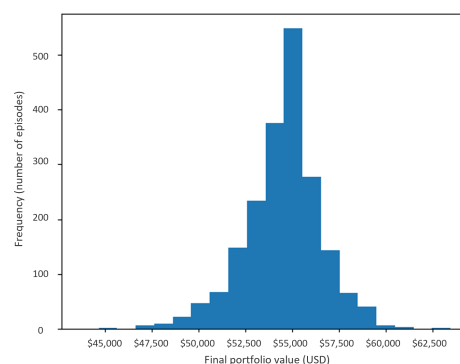


Figure 5. Histogram of final portfolio values across all test episodes for the DQN–Sharpe Ratio Maximization framework on the emerging-market ETF test dataset.

To assess the effectiveness of the proposed attention-based state-space construction, we quantitatively compare the attention-based state-space construction with traditional preprocessing methods by evaluating their downstream out-of-sample trading performance under an identical DQN architecture. The out-of-sample test results are as follows:

- Attention-based: Final portfolio value \$45,448, mean daily return 0.3151%, annualized volatility 25.35%, annual Sharpe 3.1314, maximum drawdown 11.32%.
- Kalman-based: Final portfolio value \$24,105, mean daily return 0.19%, annualized volatility 26.84%, annual Sharpe 1.7835, maximum drawdown 14.73%.

- Wavelet-based: Final portfolio value \$7806, mean daily return -0.0353% , annualized volatility 26.75% , annual Sharpe -0.3323 , maximum drawdown 32.02% .

As the results demonstrate, the attention-based approach consistently outperforms traditional techniques such as wavelet and Kalman-based preprocessing across all economic metrics, including final portfolio value, Sharpe ratio, and drawdown. The key reason for this performance gap lies in the mismatch between traditional feature-extraction objectives and reinforcement learning requirements. Classical methods such as wavelet transforms and Kalman filtering are designed to denoise signals under strong assumptions (e.g., stationarity, linear dynamics, or predefined frequency bands) and operate in a task-agnostic manner. As a result, they may suppress regime-dependent, cross-asset, or temporally asymmetric patterns that are not noise from a signal-processing perspective but are highly informative for sequential decision-making and portfolio allocation.

4.6. Pure-Optimization Portfolios

To compare our DQN driven signals with traditional optimisation approaches we also simulated so-called pure-optimisation portfolios, which calculate an optimal weight vector across the entire asset universe and then rebalance the portfolio at the close price using those weights. In other words, we are isolating the power of each optimization rule individually with no prior signal screening or RL filtering and we are simply examining how much additional value the RL signals add. Starting with an initial \$10,000, the cumulative final investment values for each optimization method are summarized in Table 6. This comparison shows how much pure optimization alone can achieve, and demonstrates the added value of combining RL-derived signals with portfolio optimization.

Table 6. Final portfolio values for pure optimization methods (without RL) on the emerging-market ETF test dataset (2023–2024).

Portfolio Optimization Method	Final Portfolio Value
Naïve 1 / N (Equal-Weight)	\$10,694
Risk Parity	\$11,686
Global Minimum Variance	\$11,564
Markowitz Mean Variance	\$11,166
Sharpe Ratio Maximization	\$14,502

4.7. Robustness Check

To assess robustness beyond extreme volatility periods, the entire experimental pipeline of the hybrid framework was also applied to the same assets over January 2009 to December 2018, confirming consistent performance of the hybrid framework in a non-crisis regime.

We used the same performance metrics that measure the profitability of the model and its ability to manage risk. Based on Table 7, the hybrid DQN frameworks continue to outperform the basic DQN strategy across all key metrics, demonstrating the robustness of our approach in a non-crisis regime. The basic DQN achieves a cumulative return of 60.4% with an annual volatility of 25.95% and a Sharpe ratio of 1.045 , suffering a maximum drawdown of 19.69% . On the other hand, the combination of the basic DQN with either the Markowitz Mean–Variance or Sharpe Ratio Maximization allocations again yields the strongest results, mirroring our 2023–2024 findings. Overall, the consistency of these improvements across different market regimes highlights the value of our hybrid DQN-optimization architecture.

Table 7. Performance metrics for the emerging-market ETF test dataset in the non-crisis regime (2017–2018).

Hybrid DQN + Portfolio Optimization	Cumulative Return	Volatility	Sharpe Ratio	Maximum Drawdown (%)
Basic DQN	0.604	0.2595	1.0450	19.69
DQN + Naïve 1/ <i>N</i> (Equal-Weight)	0.745	0.2418	1.2780	21.26
DQN + Risk Parity	0.716	0.2402	1.2469	22.05
DQN + Global Minimum Variance	0.761	0.2264	1.3697	17.50
DQN + Markowitz Mean Variance	0.829	0.2765	1.2373	19.10
DQN + Sharpe Ratio Maximization	0.921	0.2751	1.3289	17.80

Table 8 shows the ending portfolio values at the end of the test dataset (computed by compounding their cumulative returns over 2017 and 2018). Based on the results, the integration of the DQN with Sharpe Ratio Maximization strategy once again outperformed all others, achieving the highest terminal value of \$19,206. It is followed by the Markowitz Mean Variance approach at \$18,297 and the Global Minimum Variance at \$17,611.

Table 8. Final portfolio values for combinations of DQN and portfolio optimization methods on the emerging-market ETF test dataset during the non-crisis regime (2017–2018).

Hybrid DQN + Portfolio Optimization	Final Portfolio Value (Average)	% vs. Basic DQN
Basic DQN	\$16,044	—
DQN + Naïve 1/ <i>N</i> (Equal-Weight)	\$17,446	+8.7%
DQN + Risk Parity	\$17,156	+6.9%
DQN + Global Minimum Variance	\$17,611	+9.8%
DQN + Markowitz Mean Variance	\$18,297	+14.0%
DQN + Sharpe Ratio Maximization	\$19,206	+19.7%

Although the main focus of our analysis is on three emerging-market ETFs, we have implemented multiple experiments using different asset types, timeframes, and market conditions to assess the robustness of the hybrid framework. So, the comparison of our hybrid framework with the basic DQN has been done on three individual stocks from different sectors in the Dow Jones Industrial Average including AAPL (technology), IBM (information technology/services), and AMGN (biotechnology) from November 2011 to December 2021, and the test period covers the final two years (2020–2021). In the U.S. equities experiment, we did not reuse the ETF-tuned hyperparameters. Instead, we repeated the same time-ordered tuning protocol as in the ETF setting, and we fixed the selected configuration before evaluating performance once on the held-out test period. As shown in Table 9, the results confirm the effectiveness of our framework across different types of financial assets. Once again, the hybrid DQN with Sharpe Ratio Maximization outperforms all other configurations with a final investment value of \$82,574, the highest Sharpe ratio, and the lowest maximum drawdown. The final average portfolio value reached \$82,574, significantly higher than the \$54,994 achieved by the basic DQN. These findings confirm the generalizability of the proposed hybrid approach across different asset types (stocks and ETFs) and various market conditions.

Table 9. Performance metrics for the U.S. equities test dataset (2020–2021).

Hybrid DQN + Portfolio Optimization	Volatility	Sharpe Ratio	Maximum Drawdown (%)	Final Portfolio Value (Average)
Basic DQN	0.2805	3.1217	11.58	\$54,994
DQN + Naïve 1/ <i>N</i> (Equal-Weight)	0.2712	3.3101	16.08	\$56,716
DQN + Risk Parity	0.2510	3.8457	8.68	\$65,752
DQN + Global Minimum Variance	0.2285	4.0995	7.71	\$62,784
DQN + Markowitz Mean Variance	0.2535	3.8061	9.11	\$65,631
DQN + Sharpe Ratio Maximization	0.2469	4.3611	7.08	\$82,574

5. Discussion

In our experimental design, the comparison between the DQN and the four technical indicator strategies (MA, MACD, RSI, Bollinger Bands) was intentionally constructed to isolate the selection effect of the learned signal. Concretely, both the basic DQN and each technical rule produce daily buy/sell signals on the same asset universe and no portfolio optimization is applied in this phase. In other words, when multiple buy signals occur on the same day, we apply the same sequential allocation rule to all policies. Under these equal conditions, the basic DQN attains a final portfolio value of \$45,448 on the test set, while the best technical rule (MA) reaches only \$19,642. These results indicate that a large portion of the hybrid framework’s outperformance stems from the DQN’s ability to identify superior investable assets compared to fixed technical rules.

To quantify the allocation effect, we compared each hybrid configuration (DQN + optimizer) directly to the basic DQN. In these follow up experiments, the assets flagged by the DQN are passed to an optimizer (1/*N*, GMV, Risk Parity, Markowitz MV, or Sharpe ratio maximization) and a weight vector is computed for that day’s shortlist. Holding the DQN selection fixed, every optimizer produced a measurable incremental improvement in terminal wealth and risk adjusted performance. Therefore, while the DQN supplies the crucial signal that determines which assets enter the investable universe, the optimizer meaningfully refines how capital is allocated among those assets, thereby improving weight choice and delivering more consistent, risk-adjusted outcomes. Operationally, we decompose the hybrid improvement into a selection component and an allocation component to make these contributions transparent.

6. Limitations

While the selected ETFs and U.S. stocks provide a compact and interpretable testbed with high liquidity, we acknowledge that restricting the empirical evaluation to three assets limits the generality of our quantitative findings. A small asset universe reduces cross-sectional variation and may understate practical challenges that emerge in larger universes. We therefore view the present study as a proof-of-concept for this hybrid architecture while validating the approach on broader equity universes is important directions for future work.

Also, our robustness checks demonstrate the hybrid framework’s effectiveness across multiple periods and asset types; however, we acknowledge that regime-adaptive portfolio optimization could offer additional risk control (for instance, favoring GMV during stress episodes). Designing a reliable regime detector and rigorously validating switching rules involves substantial additional modelling and is reserved for future research.

7. Conclusions

In this paper, a novel approach to multi-stock trading is proposed, integrating the control capabilities of Deep Q-Networks (DQN) with the precision of portfolio optimization methods. The approach addresses critical challenges in reinforcement learning applications to financial trading, such as handling noisy stock data and managing the complexity of continuous action spaces. The use of an attention-based neural network to preprocess stock price data helped mitigate the effects of market noise, enhancing the agent's ability to make informed decisions. The proposed framework decomposes the problem of stock selection from the problem of capital allocation, making the calculation more efficient without sacrificing the flexibility of the strategies. The use of portfolio optimization techniques helps the system to manage the weight of the assets and make changes in response to the market environment.

Initial experiments show that even the basic DQN model with a sequential allocation strategy outperforms common technical indicators such as Moving Average, MACD, Bollinger Bands, and RSI in our test settings. Moreover, integrating any of the portfolio optimization models examined in this paper into the basic DQN yields improvements across most performance metrics. Among the hybrid frameworks, DQN combined with Sharpe Ratio Maximization delivers the most consistent gains across multiple evaluation criteria. Robustness checks across different assets (emerging-market ETFs and individual U.S. equities), sectors, and time periods indicate that these improvements persist across distinct market environments. Importantly, the proposed hybrid design is motivated by tractability. In other words, the DQN handles a discrete asset-selection decision, while the optimizer solves the continuous capital-allocation problem over the selected assets. It is worth mentioning that the empirical evaluation is not restricted to ETFs but uses emerging-market ETFs as a deliberate methodological testbed, chosen for their liquidity, price discovery, and sectoral and geographic diversity. Robustness is demonstrated by evaluating alternative historical periods and extending the framework to individual U.S. equities, showing that the model learns transferable patterns across asset types, regions, sectors, and market regimes. While scaling to larger asset universes is currently constrained by the combinatorial growth of the discrete action space inherent to the DQN formulation, the selected experimental setting ensures stable learning and clear interpretation of results. This reflects a trade-off between scalability and experimental clarity rather than a fundamental limitation, and future work will address large-scale portfolios using RL architectures better suited for high-dimensional action spaces.

8. Future Works

Several directions could extend this study. First, the applicability and performance of the current framework could be expanded by exploring alternative deep reinforcement learning architectures. Relatedly, investigating a risk-aware reward function, including modular multi-objective reward formulations that combine return and explicit downside-risk components, could better align training with realistic investment objectives. Moreover, applying the model to a larger number of stock data across a range of sectors or markets would provide insights into the model's applicability to the increased volume and variety of stocks. Developing a regime-aware optimizer that dynamically shifts objectives or risk budgets in response to market states is a promising extension.

In addition, future work could incorporate explicit asset-pricing structure into the allocation stage, for example by using CAPM- or multifactor-based expected returns and time-varying beta forecasts. This may be particularly relevant under inflationary regimes, where beta stability can deteriorate, as documented in recent evidence on inflation-driven instability in U.S. sectoral betas (Valadkhani, 2025). Another extension is to combine

the hybrid RL–optimization pipeline with equity risk premium forecasting models (e.g., econometrically robust ERP predictors) to generate forward-looking expected returns for mean–variance or Sharpe-ratio objectives (Jurdi, 2022). Finally, while MV efficiency tests are valuable for assessing whether a portfolio lies on the efficient frontier under a given return model, the aim of this paper is to evaluate out-of-sample trading performance of RL-generated signals and hybrid allocation rules under a controlled execution protocol. Applying efficiency tests to dynamic, time-varying strategies is an interesting extension that we leave for future work.

Author Contributions: Conceptualization, S.S. and F.N.; methodology, S.S.; software, S.S.; validation, S.S. and F.N.; formal analysis, S.S.; investigation, S.S.; resources, F.N.; data curation, S.S.; writing—original draft preparation, S.S.; writing—review and editing, S.S. and F.N.; visualization, S.S.; supervision, F.N.; project administration, F.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada through the NSERC Discovery Grant RGPIN201906966.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset is available upon request. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Agarwal, P. (2023). The Gibbons, Ross, and Shanken test for portfolio efficiency: A note based on its trigonometric properties. *Mathematics*, 11(9), 2198. [\[CrossRef\]](#)
- Amihud, Y. (2002). Illiquidity and stock returns: Cross-section and time-series effects. *Journal of Financial Markets*, 5(1), 31–56. [\[CrossRef\]](#)
- Betancourt, C., & Chen, W.-H. (2021). Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. *Expert Systems with Applications*, 164, 114002. [\[CrossRef\]](#)
- Bhunia, A. K., Biswas, A., & Shaikh, A. A. (2018). Extended nondominated sorting genetic algorithm (ENSGA-II) for multi-objective optimization problem in interval environment. In *Multi-objective optimization: Evolutionary to hybrid framework* (pp. 215–241). Springer.
- Biondo, A. E., Mazzarino, L., & Pluchino, A. (2024). Trading strategies and financial performances: A simulation approach. *International Review of Financial Analysis*, 95, 103426. [\[CrossRef\]](#)
- Chandar, S. K. (2022). Convolutional neural network for stock trading using technical indicators. *Automated Software Engineering*, 29(1), 16. [\[CrossRef\]](#)
- Cui, T., Du, N., Yang, X., & Ding, S. (2024). Multi-period portfolio optimization using a deep reinforcement learning hyper-heuristic approach. *Technological Forecasting and Social Change*, 198, 122944. [\[CrossRef\]](#)
- Dastgerdi, A. K., & Mercorelli, P. (2022). Investigating the effect of noise elimination on LSTM models for financial markets prediction using Kalman filter and wavelet transform. *WSEAS Transactions on Business and Economics*, 19, 432–441. [\[CrossRef\]](#)
- DeMiguel, V., Garlappi, L., & Uppal, R. (2009). Optimal versus naive diversification: How inefficient is the 1/N portfolio strategy? *The Review of Financial Studies*, 22(5), 1915–1953. [\[CrossRef\]](#)
- Gao, Z., Gao, Y., Hu, Y., Jiang, Z., & Su, J. (2020, May 8–11). *Application of deep Q-network in portfolio management*. 2020 5th IEEE International Conference on Big Data Analytics (ICBDA) (pp. 268–275), Xiamen, China.
- Ghosh, P., Neufeld, A., & Sahoo, J. K. (2022). Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. *Finance Research Letters*, 46, 102280. [\[CrossRef\]](#)
- Huang, B., Huan, Y., Xu, L. D., Zheng, L., & Zou, Z. (2019). Automated trading systems statistical and machine learning methods and hardware implementation: A survey. *Enterprise Information Systems*, 13(1), 132–144. [\[CrossRef\]](#)
- Huang, S., Cao, L., Sun, R., Ma, T., & Liu, S. (2024). Enhancing portfolio optimization: A two-stage approach with deep learning and portfolio optimization. *Mathematics*, 12(21), 3376. [\[CrossRef\]](#)
- Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. *arXiv*, arXiv:1706.10059. [\[CrossRef\]](#)
- Jurdi, D. J. (2022). Predicting the Australian equity risk premium. *Pacific-Basin Finance Journal*, 71, 101683. [\[CrossRef\]](#)

- Lee, J., Koh, H., & Choe, H. J. (2021). Learning to trade in financial time series using high-frequency through wavelet transformation and deep reinforcement learning. *Applied Intelligence*, 51(8), 6202–6223. [\[CrossRef\]](#)
- Li, P., & Yang, L. (2025). Informed trading and return predictability in China: Research based on ensemble neural network. *Emerging Markets Finance and Trade*, 61(1), 216–240. [\[CrossRef\]](#)
- Li, Y., Liu, P., & Wang, Z. (2022). Stock trading strategies based on deep reinforcement learning. *Scientific Programming*, 2022, 4698656. [\[CrossRef\]](#)
- Li, Y., Lv, S., Liu, X., & Zhang, Q. (2022). Incorporating transformers and attention networks for stock movement prediction. *Complexity*, 2022, 7739087. [\[CrossRef\]](#)
- Li, Y., Ni, P., & Chang, V. (2020). Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing*, 102(6), 1305–1322. [\[CrossRef\]](#)
- Liang, D., Xu, Y., Hu, Y., & Du, Q. (2023). Intraday return forecasts and high-frequency trading of stock index futures: A hybrid wavelet-deep learning approach. *Emerging Markets Finance and Trade*, 59(7), 2118–2128. [\[CrossRef\]](#)
- Liu, P., Zhang, Y., Bao, F., Yao, X., & Zhang, C. (2023). Multi-type data fusion framework based on deep reinforcement learning for algorithmic trading. *Applied Intelligence*, 53(2), 1683–1706. [\[CrossRef\]](#)
- Liu, Z. (2023, November 18–20). *Reinforcement learning in portfolio management with sharpe ratio rewarding based framework*. 4th International Conference on Economic Management and Model Engineering (ICEMME 2022), Nanjing, China.
- Lv, D., Yuan, S., Li, M., & Xiang, Y. (2019). An empirical study of machine learning algorithms for stock daily trading strategy. *Mathematical Problems in Engineering*, 2019, 7816154. [\[CrossRef\]](#)
- Majumder, S., Kar, S., & Pal, T. (2018). Mean-entropy model of uncertain portfolio selection problem. In *Multi-objective optimization: Evolutionary to hybrid framework* (pp. 25–54). Springer.
- Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, 7(1), 71–91. [\[PubMed\]](#)
- Markowitz, H. M., & Todd, G. P. (2000). *Mean-variance analysis in portfolio choice and capital markets*. John Wiley & Sons.
- Merton, R. C. (1980). On estimating the expected return on the market: An exploratory investigation. *Journal of Financial Economics*, 8(4), 323–361. [\[CrossRef\]](#)
- Roncalli, T. (2013). *Introduction to risk parity and budgeting*. CRC Press.
- Seada, H., & Deb, K. (2018). Non-dominated sorting based multi-/many-objective optimization: Two decades of research and application. In *Multi-objective optimization: Evolutionary to hybrid framework* (pp. 1–24). Springer.
- Sharpe, W. F. (1998). The sharpe ratio. *Streetwise—the Best of the Journal of Portfolio Management*, 3(3), 169–185. [\[CrossRef\]](#)
- Shavandi, A., & Khedmati, M. (2022). A multi-agent deep reinforcement learning framework for algorithmic trading in financial markets. *Expert Systems with Applications*, 208, 118124. [\[CrossRef\]](#)
- Valadkhani, A. (2025). Inflation-driven instability in US sectoral betas. *Journal of Asset Management*, 26(5), 506–513. [\[CrossRef\]](#)
- Wang, C.-H., Zeng, Y., & Yuan, J. (2024). Two-stage stock portfolio optimization based on AI-powered price prediction and mean-CVaR models. *Expert Systems with Applications*, 255, 124555. [\[CrossRef\]](#)
- Wei, X., Lei, B., Ouyang, H., & Wu, Q. (2020). Stock index prices prediction via temporal pattern attention and long-short-term memory. *Advances in Multimedia*, 2020, 8831893. [\[CrossRef\]](#)
- Wu, L., Gao, D., Su, W., Liang, D., & Du, Q. (2023). A trading strategy based on analysts' industry analyses—Evidence from textual analyses of analyst reports in chinese stock market. *Emerging Markets Finance and Trade*, 59(10), 3378–3389. [\[CrossRef\]](#)
- Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., & Fujita, H. (2020). Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, 538, 142–158. [\[CrossRef\]](#)
- Yang, S. (2023). Deep reinforcement learning for portfolio management. *Knowledge-Based Systems*, 278, 110905. [\[CrossRef\]](#)

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.