Iranian Journal of Fuzzy Systems

Volume 22, Number 2, (2025), pp. 187-204

Original paper

University of Sistan and Baluchestan

Integrating fuzzy logic with deep reinforcement learning to enhance financial portfolio management

M. Rezaei, [1] and H. Nezamabadi-pour, [2]

[1,2] *Intelligent Data Processing Laboratory (IDPL), Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran.*

mo.rezaei@eng.uk.ac.ir, nezam@uk.ac.ir

Abstract

Portfolio management is a challenging task due to the uncertainty and volatility in financial markets, making precise asset allocation and return maximization difficult. This paper presents a novel deep reinforcement learning (DRL) approach enhanced with fuzzy trend indicators to improve portfolio decision-making. The model was developed using a DRL framework, where a convolutional neural network (CNN)-based policy network learns to optimize asset allocations through interactions with the market. Fuzzy trend indicators are incorporated as additional input features, enabling the model to better capture market uncertainties and ambiguous trends. By providing a more flexible representation of market conditions, fuzzy trend indicators allow the model to dynamically adjust portfolio allocations in response to changing trends, leading to more precise asset allocation decisions and enhanced portfolio performance. The proposed model was trained and evaluated on historical stock data from the Brazilian stock market, covering the period from 2011 to 2020. The dataset includes daily high, low, and closing prices, ensuring a strong foundation for model training and validation. Experimental results show that the fuzzy-enhanced model outperforms some state-of-the-art strategies in terms of both returns and adaptability to volatile market conditions.

Keywords: Portfolio management, deep reinforcement learning, risk management, fuzzy logic.

1 Introduction

Portfolio management refers to the systematic process of continuously constructing and reallocating a portfolio, where a mixture of assets, including stocks, bonds, and other securities, is invested to achieve special long-term financial goals [34]. The core objective of portfolio management is to establish the optimal weights of assets by balancing the overall risk and potential return of the portfolio according to the investor's risk tolerance and investment horizon [39]. The weights assigned to each asset in a portfolio determine the proportion of the total investment allocated to it. The allocation process is recognized as a trading action [3].

Financial markets are highly volatile and uncertain, making portfolio management a challenging task [9]. Traditional methods often rely on hand-crafted features like technical indicators to extract patterns from price data [24]. These approaches use fixed mathematical rules to balance risk and return, requiring constant manual adjustments to stay effective [55]. However, their rule-based nature limits adaptability in dynamic environments. Moreover, financial markets rarely follow strictly bullish or bearish trends—uncertainty and intermediate market states often lead to misclassification. As a result, traditional models struggle to adjust to sudden shifts, increasing the risk of poor investment decisions [12].

Unlike traditional approaches, Deep Neural Networks (DNNs), such as Recurrent Neural Networks (RNN) [41], Long-Short Term Memory (LSTM) [15], Convolutional Neural Networks (CNN) [22], and Transformers [51], automatically learn hierarchical representations from raw financial data. This process eliminates the need for manual feature

Corresponding Author: M. Rezaei
Received: October 2024; Revised: March 2025; Accepted: May 2025.
https://doi.org/10.22111/ijfs.2025.50135.8847

engineering and improving model adaptability to different market conditions strategies [17, 19, 30, 57, 61, 62]. While DNNs enhance feature extraction, do not explicitly model uncertainty [9]. To bridge this gap, fuzzy logic [60], provides a more flexible approach by modeling uncertainty explicitly, allowing the system to handle ambiguous and intermediate market states more effectively [12]. By integrating DNNs with fuzzy logic, portfolio management models can both automate feature extraction and enhance decision-making under uncertainty, leading to more robust and adaptable strategies [9].

There are two general approaches for constructing a portfolio using DNNs: two-step and single-step methods [3]. In the two-step approach, a predictive model is first trained using historical asset data to forecast future prices, either as a classification task (predicting bullish, bearish, or oscillating trends) [5, 56], or a regression task (predicting exact prices) [10, 32, 59]. The predictions are then used by a trading agent to construct a portfolio [32]. While simple, this approach has drawbacks: it does not directly execute trades but requires additional heuristic rules for decision-making [3, 13], and its effectiveness depends heavily on price prediction accuracy [19]. Moreover, reducing prediction errors does not always lead to profit maximization, resulting in suboptimal performance [21].

To address these limitations, the single-step approach directly selects portfolio allocations without forecasting asset prices [37]. Instead, DNNs learn to optimize asset weights from historical prices, often using reinforcement learning (RL) techniques [40]. RL, introduced by Sutton and Barto [49], enables agents to learn optimal decision-making by interacting with their environment. It has demonstrated remarkable success, particularly in game-playing, such as AlphaGo and Atari games, where agents learned strategies through trial and error [36, 45].

Inspired by these successes, RL has been applied to financial markets, where trading agents dynamically adjust portfolio allocations based on market conditions, similar to how human investors refine their strategies over time [1, 2, 18, 19, 44]. DRL integrates DNNs with RL, combining deep learning's feature extraction capabilities with RL's sequential decision-making framework [16, 38]. In DRL-based portfolio selection, a trading agent interacts with the market, receives rewards based on accumulated profits, and updates its decision policy to maximize long-term returns [1, 2].

To evaluate the impact of fuzzy indicators on DRL-based portfolio selection, this study adopts the Ensemble of Identical Independent Evaluators (EIIE) framework, a well-established DRL-based model [19]. EIIE was selected for three key reasons. First, its open-source implementation ensures reproducibility [63], whereas many advanced DRL models lack publicly available code, making replication difficult. Second, EIIE is computationally efficient and interpretable, unlike transformer-based models, which require large-scale datasets and extensive computational resources [46, 48, 61]. Finally, using a simpler, well-validated framework allows us to isolate and analyze the effect of fuzzy indicators without interference from complex architectures. While this study focuses on validating the role of fuzzy logic in portfolio management, future research will extend this approach to more advanced models, such as transformers, to assess its broader applicability.

The proposed fuzzy-enhanced EIIE (F-EIIE) model offers several advantages over existing deep learning-based portfolio selection approaches. First, unlike conventional DRL models that rely on price-based features, our method explicitly models uncertainty through fuzzy logic, improving decision-making in ambiguous market conditions [12]. Second, while transformer-based models provide strong sequence modeling capabilities, they require extensive computational resources and large-scale datasets, making them less practical for small-scale financial applications. Our approach, built on the computationally efficient EIIE framework, maintains strong performance while remaining suitable for resource-constrained environments [19]. Finally, by combining DRL with fuzzy logic, our model enhances portfolio adaptability, allowing for more dynamic asset allocation in response to market fluctuations [9]. These improvements demonstrate that integrating fuzzy indicators into DRL enhances portfolio management by improving robustness, efficiency, and decision-making under uncertainty. Our contributions include:

- The integration of fuzzy logic indicators with price features in a deep reinforcement learning framework.
- A computationally efficient method for capturing market uncertainty and ambiguous trends without the high parameter demands of more complex models.
- Demonstrating that this integration improves decision-making and portfolio allocation in dynamic market environments, with EIIE's simplified structure proving effective in dynamic market conditions, especially for resource-constrained trading environments.

The structure of the remainder of the paper is as follows: In Section 2, the studies and drawbacks are discussed. Section 3 presents the background of the study, detailing the formalization of the portfolio selection problem and the reinforcement learning. In Section 4, the proposed method is introduced. In Section 5, the experimental results on the stock dataset are presented, and Section 6 summarizes the study in conclusions and some possible future work.

2 Related works

In the context of portfolio selection, the time horizon represents the period within which the portfolio manager plans and executes investment strategies [10]. Based on the different time horizons over which investment decisions are made, portfolio selection approaches can be categorized into single-period and multi-period [24].

In single-period portfolio selection, the portfolio manager allocates assets across various investment options based on the expected return and risk characteristics for a single period. Markowitz's mean-variance (MV) model, also known as Modern Portfolio Theory (MPT), was developed for single-period investment [34]. In the MV model, the process of selecting a portfolio is guided by balancing expected return (mean) and risk (variance), constructing the portfolio to align with an investor's specific risk and return preferences [24].

Alternatively, multi-period portfolio selection approaches consider the performance of a portfolio over multiple periods, rather than just a single period. This approach is especially relevant for investors aiming to grow their wealth over the long term [24]. Capital Growth Theory, originally derived from information theory, guides investment decisions over multiple periods [20]. As market conditions change, the portfolio can be rebalanced to optimize for the highest possible growth rate, taking into consideration the compounding effect of returns over time. Both MV and Capital Growth Theory address the challenge of optimal portfolio selection; however, the second is specifically designed for dynamic situations involving multiple investment periods [62]. This sequential (online) scenario is the focus of this paper, which inherently involves a series of consecutive periods of investment decision-making. Here, the portfolio undergoes a process of rebalancing to predetermined allocations at the end of each trading period [7]. The capital growth theory is implemented through two main approaches: online learning methods and cutting-edge techniques based on RL [62].

2.1 Online portfolio selection

Online portfolio selection (OLPS) includes four distinct strategies: "Follow-the-Winner", "Follow-the-Loser", "Pattern-Matching", and "Meta-Learning" [24]. The "Follow-the-Winner" strategy involves investing more in assets that have performed well in the past, under the assumption that they will continue to perform well. The Universal Portfolio (UP) algorithm [7], and the Exponential Gradient (EG) [14] are two well-known strategies in which the past performance is used to allocate more weight to the winning assets.

Contrary to the "Follow-the-Winner" strategy, the "Follow-the-Loser" strategy involves investing more in assets that have performed poorly in the recent past. This strategy derives from the principle of mean reversion, which refers to the financial theory that asset prices and returns eventually revert to their long-term mean or average [24]. The Anti-Correlation (Anticor) algorithm [4], Passive Aggressive Mean Reversion (PAMR) [26], Confidence Weighted Mean Reversion (CWMR) [25], and Online Moving Average Reversion (OLMAR) [23], are the most famous algorithms in the "Follow-the-Loser" category.

"Pattern-matching" based strategies utilize non-parametric prediction methods to identify recurring patterns in historical market data and make investment decisions based on these patterns [11]. The three online portfolio selection strategies mentioned earlier, utilize a single investment strategy. They are designed to capitalize on specific market conditions or behaviors. However, they might not perform consistently across different market scenarios. On the other hand, Meta-Learning methods combine multiple strategies to adapt to different market conditions [62]. They analyze the performance of various investment strategies over time and combine them to create a more robust strategy. Aggregating Algorithms (AA) [52], Online Gradient Updates (OGU), and Online Newton Updates (ONU) [8], fall into this category.

Despite their theoretical appeal, OLPS strategies have notable limitations. The "Follow-the-Winner" strategy assumes that past winners will continue to perform well, which often fails in highly volatile markets where trends reverse unexpectedly. Conversely, the "Follow-the-Loser" strategy, based on mean reversion, struggles during prolonged market trends where asset prices do not revert to historical averages as expected. Pattern-matching strategies depend heavily on historical price patterns, which may not always generalize to future market conditions. Similarly, while Meta-Learning approaches improve adaptability by combining multiple strategies, they still rely on historical performance, making them vulnerable to sudden structural shifts in the market [24]. These limitations highlight the need for a more dynamic, learning-based approach, such as deep reinforcement learning, which can adjust its strategies based on real-time market feedback rather than fixed historical assumptions.

2.2 Reinforcement learning-based portfolio selection

The use of hand-crafted features, such as technical indicators in online portfolio selection methods, often leads to unsatisfactory outcomes due to their insufficient ability to model the sequential price features [9]. In addition, many

of them don't consider transaction costs, which typically arise from commission fees. Ignoring these costs can create a discrepancy between the expected and actual performance of a portfolio [61]. As a result, researchers have suggested employing DRL-based methods for developing portfolio trading strategies and directly optimizing the utility functions that represent the preferences of investors, often in terms of risk and return [47, 57, 61, 62].

Jiang et al. developed a framework to address the portfolio selection problem by utilizing deep DRL to understand market behavior and execute trading decisions. They introduced the Ensemble of Identical Independent Evaluators (EIIE) architecture to process extensive financial data and extract the price features of each asset separately. The three types of DNNs, including CNN, RNN, and LSTM have been utilized to build up the policy network. The Deterministic Policy Gradient (DPG) algorithm has been employed for model optimization [19]. Many studies built upon their framework for applying DRL to the task of portfolio selection. Liang et al. utilize three commonly used RL algorithms, DDPG, Proximal Policy Optimization (PPO), and Policy Gradient (PG), in portfolio selection on the Chinese stock market [27].

Xu et al. developed a novel policy network that is utilized by transformers. They used a self-attention mechanism to effectively process and model sequential information present in asset prices, leading to more accurate assessments of future trends or better decision-making in trading strategies [57]. Zhang et al. introduced a new portfolio selection strategy that is sensitive to costs. This strategy is designed to simultaneously process the price sequential information and asset correlation information through an innovative dual-stream network architecture [61].

Zhao et al. introduced a new asset investment policy network that is based on the self-attention mechanism. This mechanism enables the policy network to understand and model the correlations between different assets in a portfolio by focusing on the most relevant features extracted from the price data of each asset. The training of the policy network is done using a DPG algorithm [62].

Hao et al. proposed a DRL-based asset allocation approach that integrates fuzzy logic to handle market uncertainty. Their model uses a three-dimensional fuzzy vector alongside traditional market features, which are processed by five DRL algorithms: A2C, TRPO, ACKTR, PPO, and DDPG. Each algorithm has unique advantages [12]. A2C improves stability by estimating policy and value separately [35]. TRPO enhances training reliability through constrained updates [42], and PPO simplifies optimization with a clipping mechanism that balances exploration and exploitation, making it widely used in financial decision-making [43]. ACKTR enhances A2C by increasing sample efficiency using second-order optimization [54], while DDPG, an off-policy method, is particularly effective for continuous asset allocation [28]. By combining these methods, their approach aims to leverage individual strengths while compensating for weaknesses, enhancing decision-making in portfolio management [12].

While DRL-based portfolio selection models have demonstrated promising results, they also come with notable challenges. Many existing approaches rely primarily on historical price and technical indicators, limiting their ability to interpret market uncertainty. Transformers, while powerful for sequential modeling, require extensive computational resources and large-scale datasets, making them impractical for small-scale financial applications.

The proposed approach addresses key limitations in existing DRL-based portfolio selection methods which struggle with uncertainty modeling and adaptability due to their reliance on price-based features and deterministic decision-making. Prior studies have used DRL architectures for portfolio management, but these models often fail to capture uncertainty, leading to suboptimal asset allocation in volatile conditions.

This paper introduces a fuzzy logic-enhanced DRL method that explicitly models uncertainty, improving decision-making in highly volatile financial markets. Unlike standard DRL approaches, which make fixed-rule trading decisions based only on historical price trends our fuzzy-enhanced model dynamically incorporates market trend indicators, allowing for greater flexibility. Furthermore, by leveraging the efficiency of EIIE, our approach remains computationally accessible, making it suitable for resource-constrained conditions. The key innovations of this work are:

- A unique combination of fuzzy logic and DRL, leveraging both to capture not only price trends but also broader market dynamics, improving adaptability in uncertain conditions.
- Integration of fuzzy trend indicators alongside price features (high, low, close) to enhance market state detection, addressing limitations of existing DRL models that rely solely on price-based inputs.
- A computationally efficient and scalable solution, ensuring practical applicability for individual traders and smaller financial institutions, where complex models may be impractical.

3 Background

In this section, two key areas are investigated: firstly, a portfolio selection process, and secondly, reinforcement learning. This exploration aims to provide a comprehensive understanding of how these advanced computational techniques can

be leveraged for strategic financial planning and execution.

3.1 Portfolio selection process

The theoretical framework for addressing the portfolio selection problem over a financial market spanning a period of n timesteps is outlined as follows:

Assume there are $m+1$ assets, including one cash asset (usually considered risk-free) and m risk assets (such as stocks, bonds, etc.). For each timestep t, the matrix $\mathbf{x}_t \in \mathbb{R}_+^{m\times d}$ represents the prices of all m risk assets. Each row $\mathbf{x}_{t,i} \in \mathbb{R}_+^d$ indicates the features of asset i, and d is the number of different features considered for each asset. In this work, the problem feature space is constructed from three price-related features consisting of high (v_t^h), low (v_t^l), and closing (v_t^c) prices, and three fuzzy degrees ($v_t^{f_1}$, $v_t^{f_2}$, $v_t^{f_3}$), resulting in a total of six dimensions, denoted as $d=6$. The data series, with the length of k, is denoted by $\mathbf{X}_t = \{\mathbf{x}_{t-k}, \ldots, \mathbf{x}_{t-1}\} \in \mathbb{R}_+^{k\times m\times d}$, which captures the historical prices leading up to the current timestep. The relative price at time t, $\mathbf{y}_t \in \mathbb{R}^m$, is obtained by performing an element-wise division of the current price vector v_t^c by the previous price vector v_{t-1}^c:

$$\mathbf{y}_t := \frac{v_t^c}{v_{t-1}^c} = \left(\frac{v_{t,1}^c}{v_{t-1,1}^c}, \frac{v_{t,2}^c}{v_{t-1,2}^c}, \ldots, \frac{v_{t,m}^c}{v_{t-1,m}^c}\right)^T, \quad t = 1, 2, \ldots, n. \tag{1}$$

We also assume that the cash asset is risk-free and its value remains unchanged. By incorporating the relative cash asset into the relative price, we obtain an augmented vector where the first element represents the cash asset, set to 1, indicating no change in value. The augmented vector is represented by:

$$\hat{\mathbf{y}}_t = \left(1, \frac{v_{t,1}^c}{v_{t-1,1}^c}, \frac{v_{t,2}^c}{v_{t-1,2}^c}, \ldots, \frac{v_{t,m}^c}{v_{t-1,m}^c}\right)^T, \quad t = 1, 2, \ldots, n. \tag{2}$$

The portfolio weight vector $\mathbf{w}_{t-1} \in \mathbb{R}_+^{m+1}$ indicates the allocation of the total capital among the m risk assets and the cash asset in the portfolio at the start of the t-th timestep. Each component of this vector is a number in the range $[-1, +1]$ and the sum of components equals one ($\sum_i w_{t,i} = 1$) [19]. During the t-th timestep, the price changes result in a transition of the portfolio weight vector from $\mathbf{w}_{t-1}$ at the start to $\mathbf{w}'_t$ at the end. $\mathbf{w}'_t$ is calculated using the element-wise multiplication of $\hat{\mathbf{y}}_t$ and $\mathbf{w}_{t-1}$, normalized by the dot product of them.

$$\mathbf{w}'_t = \frac{\hat{\mathbf{y}}_t \odot \mathbf{w}_{t-1}}{\hat{\mathbf{y}}_t \cdot \mathbf{w}_{t-1}}. \tag{3}$$

In the next timestep, the portfolio weight vector $\mathbf{w}_t$ is adjusted following the strategy set by the trading policy for buying and selling assets. Transitioning from $\mathbf{w}'_t$ to $\mathbf{w}_t$ requires transaction costs. Let c_t represent the proportional transaction cost; then, the portfolio value is reduced by a factor of $\mu_t = 1 - c_t$. Consequently, the logarithmic rate of return of the portfolio for the t-th timestep can be established as follows:

$$r_t = \ln\left(\mu_t\, \hat{\mathbf{y}}_t \cdot \mathbf{w}_{t-1}\right). \tag{4}$$

The final portfolio value over n timesteps is achieved by:

$$p = p_0 \exp\left(\sum_{t=1}^{n} r_t\right) = p_0 \prod_{t=1}^{n} \left(\mu_t\, \hat{\mathbf{y}}_t \cdot \mathbf{w}_{t-1}\right). \tag{5}$$

where p_0 is the initial capital [19]. Full liquidity and no market impact are two necessary assumptions for backtesting the strategy, ensuring that the transactions can be executed immediately without influencing future market prices [62].

3.2 Markov decision process

Markov Decision Process (MDP) provides a mathematical framework representing the sequential decision-making problem in an environment with uncertainty, in which an agent has to make a series of decisions for which the outcomes are not known a priori. An MDP is described by a 4-tuple (S, A, P_a, r_a). S and A are finite sets of possible states and actions, respectively. $P_a(s, s')$ is the transition probability, which is the probability that choosing action a in state s at a given time, will result in a state s' at the subsequent time step. The transition probability is defined as:

$$P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a). \tag{6}$$

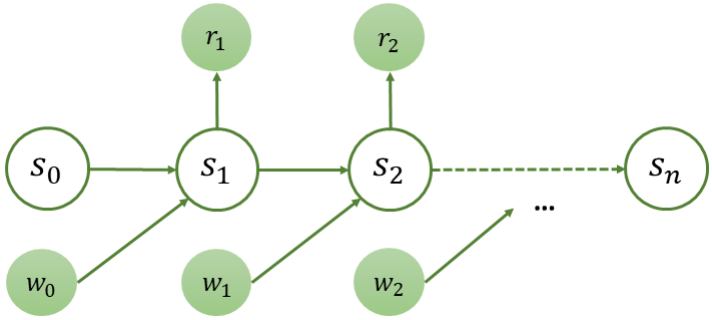

Figure 1: Transition diagram of a Markov Decision Process.

The immediate reward or the expected immediate reward, denoted by $r_a(s, s')$, is received after action a, transitions the system from state s to state s'. It is defined as:

$$r_a(s, s') = \mathbb{E}(r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'). \tag{7}$$

The reward is a numerical value that reflects the performance of the action taken. The total discounted reward accumulated at the end of an episode is called the *return*. The return at time t is calculated as:

$$R_t = \sum_{i=t}^{T} \gamma^{i-t}\, r_{a_i}(s_i, s_{i+1}), \tag{8}$$

where $\gamma \in [0, 1]$ is the discount factor and is often set close to 0.99 [33]. The policy is determined by a function $\pi(a \mid s)$ that represents the probability that action a is taken in a specific state s. The ultimate goal is to identify the optimal policy, denoted as π^*, that results in maximizing the cumulative reward [49]. The transition diagram of the MDP is shown in Figure 1.

3.3 Reinforcement learning

Reinforcement learning can solve MDPs to find the optimal policy π^*. The agent interacts with the MDP environment by following a policy, monitoring the outcomes, and modifying the policy based on the rewards obtained. This is equivalent to finding the policy that leads to the maximum state value or action value for all states. The state-value function, $V_\pi(s)$, gives the expected cumulative reward starting from state s and following policy π, and the action-value function or Q-function, denoted as $Q_\pi(s, a)$, represents the expected cumulative reward starting from state s and taking action a based on policy π [49].

The process of RL utilizes the Bellman equation to calculate the state-value and action-value functions. Following the Bellman equation's logic, the current value or action value of a state is formulated as the sum of the immediate reward and the subsequent state value or action value. The action-value function can be expressed as:

$$Q_\pi(s, a) = \mathbb{E}\left[r_t + \gamma \mathbb{E}(Q_\pi(s', a))\right]. \tag{9}$$

Various RL algorithms are employed to find the optimal policy, such as Q-learning [53] and Policy Gradient (PG) methods [50].

4 Methodology

In this section, we present our portfolio selection method, driven by DRL. We begin by introducing the notation for the DRL-based portfolio selection problem, followed by a description of the appropriate RL algorithm to solve it. Next, we detail the data preparation process and the integration of fuzzy logic for assessing market trends. The following subsections provide a detailed explanation of these components.

4.1 Deep reinforcement learning-based portfolio selection

In this section, we provide notation and clarification for solving the portfolio selection problem, using a reinforcement learning framework. This problem can be modeled as an MDP that involves making decisions on buying and selling assets at periodic time intervals within the market dynamic environment [3]. The state space of the environment at time t is composed of two distinct components: variables unaffected by the trading agent decisions and those directly influenced by the agent actions. $\mathbf{X}_t$, as defined in section 3.1, represents the first category. These external factors cannot be controlled by the agent; however, they can be observed and utilized to make informed decisions. The second component of the environment's state at time t is determined by the portfolio weight vector at the previous period, denoted by $\mathbf{w}_{t-1}$. Thus, the overall state space s_t is a concatenation of two parts, represented as:

$$s_t = (\mathbf{X}_t, \mathbf{w}_{t-1}). \tag{10}$$

The trading agent's action is defined as the portfolio weight vector where each element corresponds to a specific stock in the portfolio. A positive value means the trading agent intends to buy, while a negative value indicates a decision to sell the corresponding stock. In our trading strategy, short selling is not allowed. The trading agent aims to maximize the final portfolio value at the end of n periods, as defined by Equation 5. Since n and p_0 can not be affected by trading agent, the reward function can be expressed as the average logarithmic cumulated return [19]:

$$R := \frac{1}{n} \ln \left(\frac{p}{p_0} \right) = \frac{1}{n} \sum_{t=1}^{n} \ln \left(\mu_t \, \hat{\mathbf{y}}_t \cdot \mathbf{w}_{t-1} \right). \tag{11}$$

4.2 Deterministic policy gradient

Deterministic Policy Gradient (DPG) employed in financial portfolio management framework [19], is inspired by the Deep Deterministic Policy Gradient (DDPG), a popular method in reinforcement learning [28]. In this section, we first provide an overview of DDPG, followed by an introduction to its variant, DPG.

DDPG merges the Q-learning and PG concepts to leverage the strengths of both approaches to learn policy and value functions in high-dimensional, continuous action spaces. As an actor-critic method, DDPG includes two models, the actor for selecting actions and the critic for evaluating them. The actor network learns the policy that determines the best action to take in a given state, and the critic network estimates the value of the action output by the actor network. Throughout the training phase, the actor refines its action selection, while the critic enhances its ability to accurately estimate the value function [58].

There are significant distinctions between the DDPG and the variant used in [19], DPG, which is designed to suit the financial domain's requirements. In traditional DDPG, there is an actor-critic architecture, where the actor network selects actions and the critic network estimates the value function. In contrast, the DPG algorithm eliminates the critic network. Instead, the portfolio value is directly employed as the value function. This modification simplifies the learning process while maintaining robustness for this domain.

Also, DDPG incorporates randomness in actions through a noise parameter to promote exploration during training, helping to avoid local optima. In contrast, the DPG algorithm for portfolio management adopts a full-exploit approach, focusing exclusively on maximizing the portfolio value based on observed data without exploratory noise. In addition, while DDPG uses random sampling with a replay buffer to stabilize learning, sequential sampling is more effective in portfolio management as it captures time-dependent price variations. This approach helps the policy learn from market trends, improving decision-making by considering how past trends influence future outcomes. Figure 2 illustrates the block diagram of the DPG algorithm.

The core steps of the implemented DPG algorithm for portfolio management can be described as follows:

1. Initialize a policy network to process financial market data and generate portfolio weights, and set up a replay buffer to store experiences from each episode.
2. For each episode, proceed as follows:
   (a) Over a specified period of n = batch_size timesteps:
      i. For every timestep, define an action, which corresponds to the portfolio weights for each asset in the portfolio and store the state, action, reward, and the next state in the replay buffer.
      ii. Once a batch of timesteps is completed, sample the replay buffer sequentially.
      iii. Calculate the value function of the policy based on Equation 11.

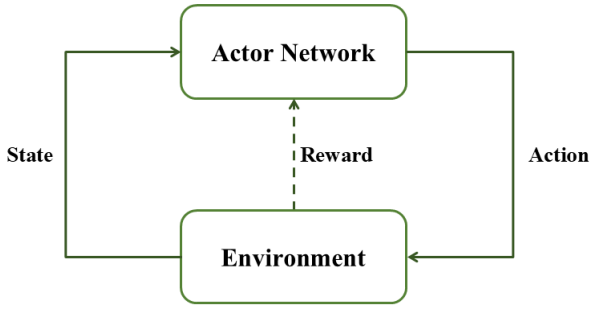

Figure 2: The block Diagram of the DPG Algorithm.

iv. Apply the gradient of the value function with respect to the policy network's parameters and update it through gradient ascent.

(b) At the end of each episode, if any remaining experiences in the replay buffer do not fit into a full batch, process these experiences similarly, ensuring that no valuable data is discarded (Jiang et al., 2017).

4.3 Fuzzy trend indicators

Financial market data is inherently uncertain due to numerous factors, such as fluctuating economic indicators, investor sentiment, and unpredictable events. While DNNs used as policy networks are highly effective for feature learning, they do not fully address the inherent uncertainties associated with financial signals. This uncertainty impacts both short-term and long-term price movements, making precise decision-making far more challenging compared to other types of sequential data, such as text or speech [9].

One possible way to address uncertainty in financial data is by incorporating fuzzy logic, which allows for modeling data with degrees of truth rather than fixed, binary categories. Fuzzy logic, introduced by Lotfi Zadeh in the 1960s, extends classical Boolean logic by allowing values to range between 0 and 1, representing partial truths. This flexibility makes it especially useful for handling complex, uncertain, or ambiguous data, as it supports more nuanced reasoning in fields like artificial intelligence, control systems, and decision-making, effectively modeling real-world scenarios with inherent vagueness [60].

For capturing the inherent ambiguity of market conditions, fuzzy rough sets are defined to capture the primary movements of stock prices: bullish, bearish, and oscillating. These fuzzy sets help classify market trends, with parameters in the fuzzy membership function either predefined based on the problem's context or learned in a fully data-driven manner. Given the complexity of financial markets, manually setting these membership functions is challenging and often suboptimal. Thus, in [9, 12, 29] data-driven methods have been proposed to directly learn the membership functions, allowing the model to adaptively capture nuanced market behaviors.

Each trading day is assigned a degree of membership to each market condition, with values ranging from 0 to 1. First, the rate of return for each stock on the j-th trading day is computed using the closing price as follows:

$$r_j = \frac{v_j^c - v_{j-1}^c}{v_{j-1}^c}. \tag{12}$$

Then, $k = 3$ fuzzy degrees are obtained as follows:

1. Select the length of the moving window T. In this case, we use $T = 60$.
2. Apply k-means clustering on the rates of returns $r_j, r_{j-1}, \ldots, r_{j-T+1}$ to group them into $k = 3$ clusters.
3. For each cluster $c = 1, 2, 3$, compute the mean M_c and standard deviation S_c of the rates of return within the cluster.
4. Compute the three fuzzy degrees f_j^c, $c = 1, 2, 3$, using the Gaussian membership function:

$$f_j^c = \exp\left(-\frac{(r_j - M_c)^2}{S_c^2}\right). \tag{13}$$

5. The computed fuzzy degrees f_j^1, f_j^2, f_j^3 for the trading day are added as fuzzy extensions to the dataset.

Figure 3 illustrates the fuzzy features connected to each trading day. In this figure, three membership functions describe the bullish, bearish, and oscillating market trends. For example, suppose we want to calculate the fuzzy degrees for a specific stock over a given trading window. We have daily returns for this stock over a 60-day period. After applying k-means clustering on these returns, we identify three clusters:

- Cluster 1 (Bullish): $M_1 = 0.02$, $S_1 = 0.01$,
- Cluster 2 (Bearish): $M_2 = -0.015$, $S_2 = 0.008$,
- Cluster 3 (Oscillating): $M_3 = 0.005$, $S_3 = 0.012$.

Now, on a particular day with $r_j = 0.01$, we can compute the fuzzy degrees f_j^1, f_j^2, f_j^3 using the Gaussian membership function:

- Bullish: $f_j^1 = \exp\left(-\frac{(0.01-0.02)^2}{0.01^2}\right) = 0.3679$,
- Bearish: $f_j^2 = \exp\left(-\frac{(0.01+0.015)^2}{0.008^2}\right) = 5.73 \times 10^{-5}$,
- Oscillating: $f_j^3 = \exp\left(-\frac{(0.01-0.005)^2}{0.012^2}\right) = 0.8405$.

This result suggests that on this day, the stock has a higher membership degree to the oscillating trend, followed by a moderate membership to the bullish trend, and almost no membership to the bearish trend. This aligns with fuzzy logic's capacity to capture partial truths, as the market condition isn't strictly categorized into a single trend but rather shows varying degrees of membership to each trend. Such a nuanced approach is particularly valuable for handling the inherent ambiguity of financial data, enabling more flexible and realistic modeling of market behavior.

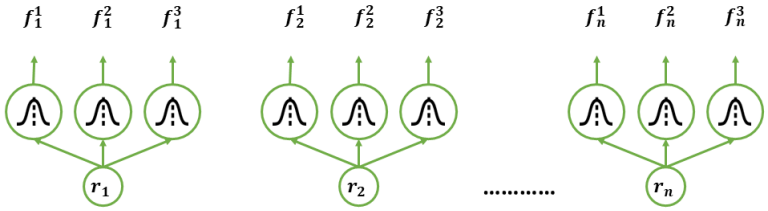

Figure 3: Overview of the calculation process for fuzzy indicators corresponding to n trading days.

4.4 Model architecture

In this section, we introduce our proposed method. The present work extends a network topology so-called Ensemble of Identical Independent Evaluators (EIIE) described in [19], by integrating fuzzy trend indicators as additional features, enhancing the model's ability to make informed portfolio decisions. In the EIIE topology, each Identical Independent Evaluators (IEE) processes the data of a single asset independently. Figure 4 depicts the structure of our proposed portfolio selection method.

As shown in Figure 4, the input layer combines the price tensor (including historical high, low, and closing data) with fuzzy degrees into a single input tensor with dimensions $6 \times m \times k$. The policy network uses CNN layers (Conv1, Conv2, Conv3) to process the input features and produce the portfolio weights. For the EIIE structure, the features are extracted in the convolutional layers through convolutional operations with kernels of height 1.

At the end of this independent processing, each IIE produces a voting score for its respective asset. These scores represent the relative attractiveness or potential of each asset as determined by the model. The inclusion of the portfolio weights from the previous period ($\mathbf{w}_{t-1}$), enables the model to account for transaction costs. This ensures that adjustments to the portfolio are based not only on current market conditions but also on historical allocations. So, $\mathbf{w}_{t-1}$ is added to the last hidden layer of the network before producing the voting scores. Then, the actor network passes the scores from all IIEs through a softmax layer to produce the final portfolio weights.

To illustrate the model's real-world application, consider an investment scenario where the trading agent manages a portfolio of five stocks over a 60-day period. The model receives historical high, low, and close prices alongside fuzzy trend indicators (bullish, bearish, oscillating) computed using a rolling window.

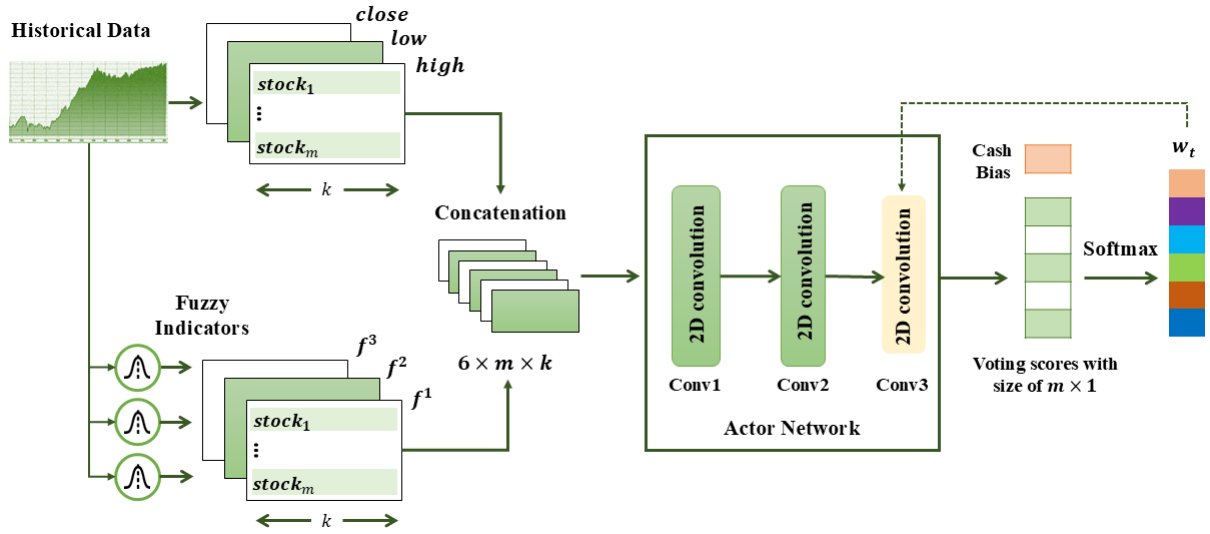

Figure 4: The block diagram of the proposed method. A price tensor with a shape of $6 \times m \times k$, representing the high, low, and close prices and three fuzzy indicators of m non-cash assets across the last k timesteps is fed into the actor network. The outputs of the network represent the updated portfolio weights. To help the agent minimize transaction costs, the previous portfolio weights are incorporated as an additional feature map before the scoring layer.

Step 1: Observing Market Conditions
On day 60, the agent analyzes the past 60 days of stock prices. Fuzzy logic assigns the following trend probabilities for a particular stock:

- Bullish: 0.65,
- Bearish: 0.10,
- Oscillating: 0.25.

Step 2: Portfolio Decision-Making
A high bullish membership suggests momentum continuation, prompting the model to increase allocation to this stock. If the oscillating membership were higher, the model would allocate conservatively, reducing exposure to market fluctuations. If bearish membership dominated, the agent would reduce allocation or exit the position to minimize losses.

Step 3: Executing the Trade
The EIIE network processes this data, adjusting portfolio weights based on predicted market conditions. Suppose that the previous weight for this stock was 20%. Based on fuzzy-enhanced predictions, the model increases allocation to 30%, anticipating an upward trend. After executing the new allocation, the agent monitors performance and updates its strategy in the next period, continuously adapting to evolving market trends.

5 Experiments

The open-source FinRL library was selected to conduct the experiments and obtain the results. FinRL is an open-source Python library designed for developing stock trading strategies in quantitative finance using DRL algorithms [31].

5.1 Data collection

The original EIIE implementation in [19] was developed for cryptocurrency portfolio selection. They constructed a cryptocurrency portfolio based on the top 11 cryptocurrencies by 30-day trading volume. However, since the specific time for this classification is not provided, replicating their method poses challenges. The FinRL framework provides an accessible implementation of EIIE using Brazilian stock market data, which we adopt in this study [63]. Using this version ensures consistency with an easily reproducible and well-documented implementation, allowing us to focus on

evaluating the impact of fuzzy indicators rather than reconfiguring the model for a different market. To adopt a similar strategy in the Brazilian stock market, we applied the following approach:

- We selected the top 10 stocks from the Brazilian stock market, ranked by their average trading volume over the past 30 days. This ensures that the most liquid and actively traded stocks are included in the portfolio.
- To maintain data consistency and accuracy, we excluded any stocks with missing data during the selected period from January 1, 2011, to December 31, 2020, which spans ten years. This filtering step ensures that the final dataset is complete, allowing for more reliable analysis without gaps that could distort results.

By focusing on liquid stocks with comprehensive data coverage, we aim to create a robust dataset for further modeling and portfolio optimization. The Brazilian stock market's high liquidity and volatility make it a suitable testbed for evaluating adaptive AI-driven investment strategies in dynamic market conditions.

Historical stock market data was gathered through the Yahoo Finance API, which provides free access to a broad array of financial information without specific usage restrictions. The API supplied daily price data for the top 10 stocks from the Brazilian stock market. We utilized the YahooDownloader class from the FinRL open-source library [31], to retrieve daily stock data. This implementation allows efficient and accurate extraction of daily stock information such as opening price, closing price, highest and lowest prices directly from the Yahoo Finance API. The final dataset includes price-based features (High, Low, Close prices) and fuzzy trend indicators (Bullish, Bearish, Oscillating) derived from fuzzy logic-based market trend analysis, providing a comprehensive representation of market conditions.

The data was preprocessed to remove missing values, standardize formats, and ensure compatibility with the DRL framework. Using real historical data provides a strong foundation for testing how well the portfolio optimization strategies work in this study. The stock market data is divided into training and trading sets. The policy network is trained using data from the training set covering the period from January 1, 2011, to December 31, 2019. Additionally, the trading set from January 1, 2020, to December 31, 2020, is used to evaluate the performance of this model.

5.2 Environment setup

As stated earlier, to effectively apply RL, it's essential to model the dynamic interactions between the agent and a task-specific environment. In the context of financial markets, the agent's objective is to trade assets over time to maximize profit. Therefore, the task-specific environment must be capable of simulating market behavior across different periods, executing the agent's buy and sell actions, and calculating the resulting gains or losses.

One of the most well-defined environments is the Portfolio Optimization Environment (POE), introduced in [6]. This environment is designed to be compatible with state-of-the-art approaches and agents, and it is integrated into the FinRL framework. POE follows the OpenAI Gym standards, making it compatible with popular RL repositories such as Stable Baselines 3 and Ray RLlib. Additionally, POE can compute various performance metrics, allowing researchers to compare different portfolio optimization methods more effectively.

5.3 Compared methods

We compare our fuzzy-enhanced EIIE (F-EIIE) methodology against a diverse range of models, including the original EIIE model using only historical price data [19], Uniformly buy and hold (UBAH) strategy, and some OLPS methods such as UP [7], EG [14], Anticor [4], PAMR [26], CWMR [25], and OLMAR [23]. This comprehensive comparison allows us to evaluate the F-EIIE model's performance across different portfolio management paradigms. Table 1 summarizes the main characteristics, advantages, and disadvantages of each model.

We selected these models based on their diversity in methodology, ranging from baseline approaches like UBAH to advanced deep reinforcement learning models. The original EIIE model serves as a baseline for evaluating the impact of fuzzy indicators, while OLPS methods represent traditional online learning strategies widely used in financial portfolio management. This selection aligns with prior research methodologies [19, 61, 62], ensuring a comprehensive and fair performance comparison.

5.4 Performance metrics

In portfolio management, several key metrics are used to evaluate the performance and risk of an investment strategy. Following [19, 57, 61, 62], we assess the effectiveness of each method using three standard performance metrics, including Accumulated Portfolio Value (APV), Sharpe Ratio (SR), and Maximum Drawdown (MDD). The APV reflects the total value of a portfolio over multiple time steps, indicating how much the portfolio has increased or decreased since its

Table 1: comparison model characteristics.

Model	Description	Advantages	Disadvantages
EIIE [19]	Original Ensemble of Identical Independent Evaluators framework using CNNs for feature extraction.	Computationally efficient, interpretable structure, proven in financial environments.	Struggles with modeling ambiguous market states and handling uncertainty.
UBAH	Uniform Buy-and-Hold strategy where assets are equally weighted and held throughout the investment period.	Simple, low transaction costs, baseline for passive investment.	Cannot adapt to changing market conditions, static allocation.
UP [7]	Universal Portfolio learns portfolio weights by tracking the best historical performer.	Theoretically optimal under certain conditions, adaptive to market trends.	Computationally intensive, sensitive to noise, slow convergence.
EG [14]	Exponential Gradient strategy adjusts portfolio weights using multiplicative updates.	Fast updates, effective in stable environments.	Limited adaptability in volatile markets.
Anticor [4]	Anti-correlation strategy based on mean reversion.	Exploits mean-reverting assets effectively.	Fails during prolonged market trends without mean reversion.
PAMR [26]	Passive Aggressive Mean Reversion strategy updates weights when portfolio deviation exceeds a threshold.	Adaptability to changes in market conditions, fast execution.	Sensitive to parameter settings, vulnerable to absence of mean reversion.
CWMR [25]	Confidence Weighted Mean Reversion adjusts based on confidence in mean reversion.	Better risk management through confidence estimation.	Complexity in parameter tuning, underperforms in trending markets.
OLMAR [23]	Online Moving Average Reversion strategy tracks moving averages to exploit mean reversion.	Enhances forecasting in volatile markets, adaptability to changes in market dynamics.	Complexity in parameter tuning, lag in rapidly shifting markets.

initial investment. APV considers both the changes in asset prices and the impact of trading costs. It is computed using Equation 5.

APV emphasizes growth without considering the associated risks. On the other hand, the SR evaluates both return and risk, adjusting returns by the portfolio's volatility. It compares the excess return over the risk-free rate to the risk taken. SR is calculated as follows:

$$\text{SR} = \frac{\text{Average}(r_t - r_f)}{\text{Standard Deviation}(r_t)}, \tag{13}$$

where r_t represents the logarithmic rate of return as defined in Equation 4, and r_f denotes the risk-free rate. A higher Sharpe Ratio indicates that the portfolio is delivering greater returns per unit of risk, making it an effective tool for comparing the efficiency of various investment strategies.

While the SR accounts for overall volatility, it treats upward and downward movements in portfolio value equally. However, downward movements are often more critical, as they represent losses and reflect the portfolio's vulnerability during market downturns. MDD specifically focuses on the worst-case loss, measuring the largest drop in a portfolio value from its peak to its lowest point over a specific period. The MDD is calculated as:

$$\text{MDD} = \frac{\text{Trough Value} - \text{Peak Value}}{\text{Peak Value}}. \tag{14}$$

In summary, Accumulated Portfolio Value measures overall growth, the Sharpe Ratio evaluates performance adjusted for risk, and Maximum Drawdown assesses the worst-case loss, making these metrics essential for comprehensive portfolio evaluation.

5.5 Parameter setting

Figure 4 illustrates the general structure of the policy network according to the guidelines in [19]. Additionally, the specific details of the F-EIIE are thoroughly outlined in Table 2 for greater clarity. We select a window length of 50 days. The initial investment amount is assumed to be \$100,000. The learning rate is set to 0.001, with a transaction cost of 0.25%. The model is trained over 40 episodes using the Adam optimizer. The cash bias term is fixed at 0, and the initial investment weights are set as $\mathbf{w}_t = [1, 0, \ldots, 0]$. Also, the results for EIIE and F-EIIE are averaged over twenty-five runs.

Table 2: Detailed Policy Network Architecture.

Architecture	**Input-Output Shape**	**Layer Information**
Conv1	$(m, k, 6) \rightarrow (m, k-2, 2)$	Activation Function: ReLU Input Channels: 6 Output Channels: 2 Kernel Size: (1, 3)
Conv2	$(m, k-2, 2) \rightarrow (m, 1, 20)$	Activation Function: ReLU Input Channels: 2 Output Channels: 20 Kernel Size: (1, 48)
Concatenation	$(m, 1, 20) \oplus (m, 1, 1) \rightarrow (m, 1, 21)$	Concatenate previous weights as an additional feature map
Conv3	$(m, 1, 21) \rightarrow (m, 1, 1)$	Activation Function: ReLU Input Channels: 21 Output Channels: 1 Kernel Size: (1, 1)
Concatenation	$(m, 1) \oplus (1, 1) \rightarrow (m+1, 1)$	Concatenate cash bias and computed weights
Softmax	$(m+1, 1) \rightarrow (m+1, 1)$	Generate portfolio weights

5.6 Results

This section presents the outcomes of various performance metrics, which are visualized through graphs and charts to provide a clear understanding of the model's performance. Table 3 presents the results of the three performance measures discussed in Section 5.4 during the testing period. Bold font is used to highlight the best results achieved.

The results in Table 3 demonstrate the effectiveness of the F-EIIE. A clear improvement is evident across multiple performance metrics when compared to the baseline EIIE and other benchmark models, such as UBAH, UP, EG, Anticor, PAMR, CWMR, and OLMAR. The F-EIIE model achieves an APV of 3.72, outperforming the original EIIE model (3.04) and the other benchmark algorithms, which demonstrate significantly lower values. This substantial improvement in APV highlights the F-EIIE model's ability to generate higher returns over the testing period, illustrating the advantage of incorporating fuzzy logic to capture nuanced market trends. The fuzzy extensions allow the model to better adapt to varying market conditions, leading to superior asset allocation and higher profitability.

The F-EIIE also excels in risk-adjusted returns, as evidenced by its SR of 2.49, which surpasses the EIIE (2.27) and other methods. This suggests that the fuzzy-enhanced model not only delivers higher returns but also manages risk more effectively. By integrating fuzzy trend indicators, F-EIIE demonstrates a greater ability to reduce portfolio volatility, resulting in a more stable performance with fewer extreme fluctuations. In addition, The F-EIIE model shows a reduction in maximum drawdown (0.31) compared to EIIE (0.32), but some benchmark models like UBAH and UP exhibit lower drawdown values (0.25).

The higher MDD observed in both F-EIIE and EIIE models compared to UBAH and OLPS methods can be explained by their inherent dynamic nature and the complexity of the market conditions they are designed to handle.

Table 3: Performance comparisons on the test dataset.

2020-01-01 to 2020-12-31	**F-EIIE**	**EIIE [19]**	**UBAH**	**UP [7]**	**EG [14]**	**Anticor [4]**	**PAMR [26]**	**CWMR [25]**	**OLMAR [23]**
APV	3.72	3.04	1.71	1.73	1.73	1.68	1.61	1.61	2.84
Sharpe Ratio	2.49	2.27	1.71	1.74	1.74	1.36	1.20	1.21	2.33
Maximum Drawdown	0.31	0.32	0.25	0.25	0.25	0.25	0.46	0.45	0.29

Unlike UBAH and OLPS methods, which follow static, predetermined rules, the F-EIIE and EIIE models employ deep reinforcement learning to adapt to constantly changing market trends. This flexibility allows the models to capture more opportunities, but it also involves a higher level of risk, particularly during periods of high volatility. As a result, the models may experience deeper drawdowns but are equipped to recover and outperform UBAH and OLPS methods in the long run, as reflected in their superior SR and overall APV.

The results of the back-testing of trading strategies based on the F-EIIE and EIIE models are shown in Figure 5, alongside comparisons with UBAH and OLPS methods. This comparison highlights the cumulative returns generated by each strategy during the testing period from January 1, 2020, to December 31, 2020. Each method's performance is evaluated against the backdrop of market fluctuations, allowing for a comprehensive assessment of how dynamic models like F-EIIE perform relative to UBAH and OLPS methods.

From the graph in Figure 5, it is evident that the F-EIIE model consistently outperforms the other methods across the entire testing period. The model demonstrates a steady upward trajectory, indicating its ability to capture market trends effectively, especially during volatile periods. In contrast, UBAH and OLPS methods fail to generate comparable returns.

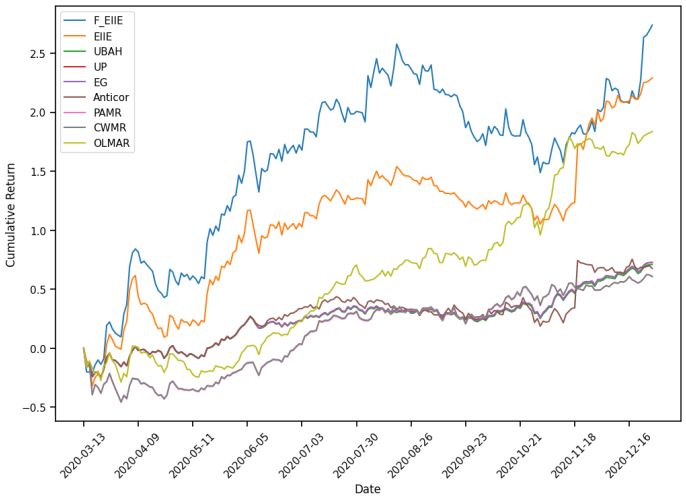

Figure 5: Back-test: 2020-01-01 to 2020-12-31. Cumulative return of the portfolio for F-EIIE, EIIE, UBAH, UP, EG, Anticor, PAMR, CWMR and OLMAR.

6 Conclusions

In conclusion, this paper presents a novel contribution to portfolio management by integrating fuzzy logic into the EIIE topology, resulting in the F-EIIE model. By incorporating fuzzy trend indicators, such as bullish, bearish, and oscillating alongside traditional price features, the model can better capture and respond to nuanced market conditions. The results demonstrate that the F-EIIE model achieves a significant improvement in cumulative returns compared to state-of-the-art methods, highlighting its ability to make more adaptive and informed investment decisions. This improvement underscores the robustness of fuzzy logic-based enhancements in deep reinforcement learning for financial markets, offering a more responsive and effective approach to portfolio management. The integration of fuzzy trends provides a middle ground between traditional and modern methods, offering a computationally efficient and adaptive framework that outperforms conventional techniques in dynamic market environments.

For future work, several directions can be explored to enhance the performance and adaptability of the F-EIIE model. Incorporating additional market features, such as macroeconomic indicators and sentiment analysis from news and social media, could improve market trend detection. Exploring advanced reinforcement learning techniques, such as multi-agent reinforcement learning, may enhance the model's generalization across different asset classes. Further research on optimizing fuzzy membership functions can refine market trend classification and improve decision-making accuracy. Additionally, extending the model to multi-objective optimization, where risk and return are balanced dynamically, could provide a more robust portfolio management approach. Another promising direction is applying fuzzy logic enhancements to more complex architectures like transformers, assessing whether these structures further

improve performance. Finally, validating the model across different financial markets, including major indices like DJIA and S&P 500, would help assess its adaptability to varying market conditions.

Acknowledgement

The authors wish to express their appreciation for several excellent suggestions for improvements in this paper made by the referees.

References

[1] R. Abdelkawy, W. M. Abdelmoez, A. Shoukry, *A synchronous deep reinforcement learning model for automated multi-stock trading*, Progress in Artificial Intelligence, **10**(1) (2021), 83-97. https://doi.org/10.1007/s13748-020-00225-z

[2] S. Almahdi, S. Y. Yang, *An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown*, Expert Systems with Applications, **87** (2017), 267-279. https://doi.org/10.1016/j.eswa.2017.06.023

[3] C. Betancourt, W. H. Chen, *Deep reinforcement learning for portfolio management of markets with a dynamic number of assets*, Expert Systems with Applications, **164** (2021), 114002. https://doi.org/10.1016/j.eswa.2020.114002

[4] A. Borodin, R. El-Yaniv, V. Gogan, *Can we learn to beat the best stock?*, Journal of Artificial Intelligence Research, **21** (2004), 579-594. https://doi.org/10.1613/jair.1336

[5] E. Chong, C. Han, F. C. Park, *Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies*, Expert Systems with Applications, **83** (2017), 187-205. https://doi.org/10.1016/j.eswa.2017.04.030

[6] C. D. S. B. Costa, A. H. R. Costa, *POE: A general portfolio optimization environment for FinRL*, Brazilian Workshop on Artificial Intelligence in Finance (BWAIF), (2023), 132-143. https://doi.org/10.5753/bwaif.2023.231144

[7] T. M. Cover, *Universal portfolios*, Mathematical Finance, **1**(1) (1991), 1-29. https://doi.org/10.1111/j.1467-9965.1991.tb00002.x

[8] P. Das, A. Banerjee, *Meta optimization and its application to portfolio selection*, Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (2011), 1163-1171. https://doi.org/10.1145/2020408.2020588

[9] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, *Deep direct reinforcement learning for financial signal representation and trading*, IEEE Transactions on Neural Networks and Learning Systems, **28**(3) (2016), 653-664. https://doi.org/10.1109/TNNLS.2016.2522401

[10] F. D. Freitas, A. F. de Souza, A. R. de Almeida, *Prediction-based portfolio optimization model using neural networks*, Neurocomputing, **72**(10-12) (2009), 2155-2170. https://doi.org/10.1016/j.neucom.2008.08.019

[11] L. Györfi, G. Lugosi, F. Udina, *Nonparametric kernel-based sequential investment strategies*, Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics, **16**(2) (2006), 337-357. https://doi.org/10.1111/j.1467-9965.2006.00274.x

[12] Z. Hao, H. Zhang, Y. Zhang, *Stock portfolio management by using fuzzy ensemble deep reinforcement learning algorithm*, Journal of Risk and Financial Management, **16**(3) (2023), 201. https://doi.org/10.3390/jrfm16030201

[13] J. B. Heaton, N. G. Polson, J. H. Witte, *Deep learning for finance: Deep portfolios*, Applied Stochastic Models in Business and Industry, **33**(1) (2017), 3-12. https://doi.org/10.1002/asmb.2209

[14] D. P. Helmbold, R. E. Schapire, Y. Singer, M. K. Warmuth, *On-line portfolio selection using multiplicative updates*, Mathematical Finance, **8**(4) (1998), 325-347. https://doi.org/10.1111/1467-9965.00058

[15] S. Hochreiter, J. Schmidhuber, *Long short-term memory*, Neural Computation, **9**(8) (1997), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[16] G. Jeong, H. Y. Kim, *Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning*, Expert Systems with Applications, **117** (2019), 125-138. https://doi.org/10.1016/j.eswa.2018.09.036

[17] Z. Jiang, J. Liang, *Cryptocurrency portfolio management with deep reinforcement learning*, 2017 Intelligent Systems Conference (IntelliSys), IEEE, (2017), 905-913. https://doi.org/10.1109/IntelliSys.2017.8324237

[18] Y. Jiang, J. Olmo, M. Atwi, *Deep reinforcement learning for portfolio selection*, Global Finance Journal, **62** (2024), 101016. https://doi.org/10.1016/j.gfj.2024.101016

[19] Z. Jiang, D. Xu, J. Liang, *A deep reinforcement learning framework for the financial portfolio management problem*, arXiv Preprint arXiv:1706.10059, (2017). https://doi.org/10.48550/arXiv.1706.10059

[20] J. L. Kelly, *A new interpretation of information rate*, The Bell System Technical Journal, **35**(4) (1956), 917-926. https://doi.org/10.1002/j.1538-7305.1956.tb03809.x

[21] P. Koratamaddi, K. Wadhwani, M. Gupta, S. G. Sanjeevi, *Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation*, Engineering Science and Technology, an International Journal, **24**(4) (2021), 848-859. https://doi.org/10.1016/j.jestch.2021.01.007

[22] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, **86**(11) (1998), 2278-2324. https://doi.org/10.1109/5.726791

[23] B. Li, S. C. Hoi, *Online portfolio selection with moving average reversion*, Proceedings of International Conference on Machine Learning, (2012), 273-280. https://doi.org/10.48550/arXiv.1206.4626

[24] B. Li, S. C. Hoi, *Online portfolio selection: A survey*, ACM Computing Surveys (CSUR), **46**(3) (2014), 1-36. https://doi.org/10.1145/2512962

[25] B. Li, S. C. Hoi, P. Zhao, V. Gopalkrishnan, *Confidence weighted mean reversion strategy for on-line portfolio selection*, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, (2011), 434-442. https://doi.org/10.1145/2435209.2435213

[26] B. Li, P. Zhao, S. C. Hoi, V. Gopalkrishnan, *PAMR: Passive aggressive mean reversion strategy for portfolio selection*, Machine Learning, **87** (2012), 221-258. https://doi.org/10.1007/s10994-012-5281-z

[27] Z. Liang, H. Chen, J. Zhu, K. Jiang, Y. Li, *Adversarial deep reinforcement learning in portfolio management*, arXiv Preprint arXiv:1808.09940, (2018). https://doi.org/10.48550/arXiv.1808.09940

[28] T. Lillicrap, *Continuous control with deep reinforcement learning*, arXiv Preprint arXiv:1509.02971, (2015). https://doi.org/10.48550/arXiv.1509.02971

[29] C. T. Lin, C. M. Yeh, S. F. Liang, J. F. Chung, N. Kumar, *Support-vector-based fuzzy neural network for pattern classification*, IEEE Transactions on Fuzzy Systems, **14**(1) (2006), 31-41. https://doi.org/10.1109/TFUZZ.2005.861604

[30] Y. Liu, D. Mikriukov, O. C. Tjahyadi, G. Li, T. R. Payne, Y. Yue, K. Siddique, K. L. Man, *Revolutionizing financial portfolio management: The non-stationary transformer's fusion of macroeconomic indicators and sentiment analysis in a deep reinforcement learning framework*, Applied Sciences, **14**(1) (2023). https://doi.org/10.3390/app14010274

[31] X. Y. Liu, H. Yang, Q. Chen, R. Zhang, L. Yang, B. Xiao, C. D. Wang, *FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance*, arXiv Preprint arXiv:2011.09607, (2020). https://doi.org/10.48550/arXiv.2011.09607

[32] Y. Ma, R. Han, W. Wang, *Prediction-based portfolio optimization models using deep neural networks*, IEEE Access, **8** (2020), 115393-115405. https://doi.org/10.1109/ACCESS.2020.3003819

[33] C. Ma, J. Zhang, J. Liu, L. Ji, F. Gao, *A parallel multi-module deep reinforcement learning algorithm for stock trading*, Neurocomputing, **449** (2021), 290-302. https://doi.org/10.1016/j.neucom.2021.04.005

[34] H. M. Markowitz, *Portfolio selection*, The Journal of Finance, **7** (1952), 77-91. https://doi.org/10.1111/j.1540-6261.1952.tb01525.x

[35] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, *Asynchronous methods for deep reinforcement learning*, International Conference on Machine Learning, (2016). https://doi.org/10.48550/arXiv.1602.01783

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, L. Antonoglou, D. Wierstra, M. Riedmiller, *Playing atari with deep reinforcement learning*, arXiv Preprint arXiv:1312.5602, (2013). https://doi.org/10.48550/arXiv.1312.5602

[37] J. Moody, M. Saffell, *Learning to trade via direct reinforcement*, IEEE Transactions on Neural Networks, **12**(4) (2001), 875-889. https://doi.org/10.1109/72.935097

[38] H. Park, M. K. Sim, D. G. Choi, *An intelligent financial portfolio trading strategy using deep Q-learning*, Expert Systems with Applications, **158** (2020), 113573. https://doi.org/10.1016/j.eswa.2020.113573

[39] M. Rasoulzadeh, S. A. Edalatpanah, M. Fallah, S. E. Najafi, *A hybrid model for choosing the optimal stock portfolio under intuitionistic fuzzy sets*, Iranian Journal of Fuzzy Systems, **21**(2) (2025), 161-179. https://doi.org/10.22111/ijfs.2024.45118.7968

[40] M. Rezaei, H. Nezamabadi-Pour, *A taxonomy of literature reviews and experimental study of deepreinforcement learning in portfolio management*, Artificial Intelligence Review, **58**(3) (2025), 1-46. https://doi.org/10.1007/s10462-024-11066-w

[41] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Learning representations by back-propagating errors*, Nature, **323**(6088) (1986), 533-536. https://doi.org/10.1038/323533a0

[42] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, *Trust region policy optimization*, International Conference on Machine Learning, (2015). https://doi.org/10.48550/arXiv.1502.05477

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, *Proximal policy optimization algorithms*, arXiv Preprint arXiv:1707.06347, (2017). https://doi.org/10.48550/arXiv.1707.06347

[44] S. Shi, J. Li, G. Li, P. Pan, Q. Chen, Q. Sun, *GPM: A graph convolutional network based reinforcement learning framework for portfolio management*, Neurocomputing, **498** (2022), 14-27. https://doi.org/10.1016/j.neucom.2022.04.105

[45] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. V. D. Riessche, T. Graepel, D. Hassabis, *Mastering the game of go without human knowledge*, Nature, **550**(7676) (2017), 354-359. https://doi.org/10.1038/nature24270

[46] F. Soleymani, E. Paquet, *Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder-DeepBreath*, Expert Systems with Applications, **156** (2020), 113456. https://doi.org/10.1016/j.eswa.2020.113456

[47] F. Soleymani, E. Paquet, *Deep graph convolutional reinforcement learning for financial portfolio management-deeppocket*, Expert Systems with Applications, **182** (2021), 115127. https://doi.org/10.1016/j.eswa.2021.115127

[48] Q. Sun, X. Wei, X. Yang, *GraphSAGE with deep reinforcement learning for financial portfolio optimization*, Expert Systems with Applications, **238** (2024), 122027. https://doi.org/10.1016/j.eswa.2023.122027

[49] R. S. Sutton, A. G. Barto, *Reinforcement learning: An introduction*, Cambridge: MIT Press, **1**(1) (2018). https://doi.org/10.1017/S0263574799271172

[50] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, *Policy gradient methods for reinforcement learning with function approximation*, Advances in Neural Information Processing Systems (NeurIPS), **13** (1999), 1057-1063. https://dl.acm.org/doi/10.5555/3009657.3009806

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, *Attention is all you need*, Advances in Neural Information Processing Systems (NeurIPS), **30** (2017), 6000-6010. https://dl.acm.org/doi/10.5555/3295222.3295349

[52] V. G. Vovk, *Aggregating strategies*, Proceedings of the Third Annual Workshop on Computational Learning Theory, https://doi.org/10.1016/B978-1-55860-146-8.50032-1

[53] C. J. Watkins, P. Dayan, *Q-learning*, Machine Learning, **8**(3-4) (1992), 279-292. https://doi.org/10.1007/BF00992698

[54] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, J. Ba, *Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation*, Advances in Neural Information Processing Systems (NeurIPS), **30** (2017), 5285-5294. https://dl.acm.org/doi/10.5555/3295222.3295280

[55] X. Wu, D. A. Ralescu, Y. Liu, *A new quadratic deviation of fuzzy random variable and its application to portfolio optimization*, Iranian Journal of Fuzzy Systems, **17**(3) (2020), 1-18. https://doi.org/10.22111/ijfs.2020.5344

[56] D. Wu, X. Wang, J. Su, B. Tang, S. Wu, *A labeling method for financial time series prediction based on trends*, Entropy, **22**(10) (2020), 1162. https://doi.org/10.3390/e22101162

[57] K. Xu, Y. Zhang, D. Ye, P. Zhao, M. Tan, *Relation-aware transformer for portfolio policy learning*, Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, (2021), 4647-4653. https://doi.org/10.24963/ijcai.2020/641

[58] H. Yang, X. Y. Liu, S. Zhong, A. Walid, *Deep reinforcement learning for automated stock trading: An ensemble strategy*, Proceedings of the First ACM International Conference on AI in Finance (ICAIF), **20** (2020). https://doi.org/10.1145/3383455.3422540

[59] H. Yun, M. Lee, Y. S. Kang, J. Seok, *Portfolio management via two-stage deep learning with a joint cost*, Expert Systems with Applications, **143** (2020), 113041. https://doi.org/10.1016/j.eswa.2019.113041

[60] L. A. Zadeh, *Fuzzy sets*, Information and Control, **8**(3) (1965), 338-353. https://doi.org/10.1016/S0019-9958%2865%2990241-X

[61] Y. Zhang, P. Zhao, B. Li, Q. Wu, J. Huang, M. Tan, *Cost-sensitive portfolio selection via deep reinforcement learning*, IEEE Transactions on Knowledge and Data Engineering, **34**(1) (2020). https://doi.org/10.1109/TKDE.2020.2979700

[62] T. Zhao, X. Ma, X. Li, C. Zhang, *Asset correlation based deep reinforcement learning for the portfolio selection*, Expert Systems with Applications, **221** (2023), 119707. https://doi.org/10.1016/j.eswa.2023.119707

[63] https://github.com/AI4Finance-Foundation/FinRL/blob/master/examples/FinRL_PortfolioOptimizationEnv_Demo.ipynb