Journal of
Risk and Financial Management

Article

Stock Portfolio Management by Using Fuzzy Ensemble Deep Reinforcement Learning Algorithm

Zheng Hao [1], Haowei Zhang [2] and Yipu Zhang [2,*]

[1] Mathematics Department, State University of New York at Oswego, Oswego, NY 13126, USA
[2] School of Energy and Electrical Engineering, Chang'an University, Xi'an 710064, China
[*] Correspondence: zyipu@chd.edu.cn

Abstract: The research objective of this article is to train a computer (agent) with market information data so it can learn trading strategies and beat the market index in stock trading without having to make any prediction on market moves. The approach assumes no trading knowledge, so the agent will only learn from conducting trading with historical data. In this work, we address this task by considering Reinforcement Learning (RL) algorithms for stock portfolio management. We first generate a three-dimension fuzzy vector to describe the current trend for each stock. Then the fuzzy terms, along with other stock market features, such as prices, volumes, and technical indicators, were used as the input for five algorithms, including Advantage Actor-Critic, Trust Region Policy Optimization, Proximal Policy Optimization, Actor-Critic Using Kronecker Factored Trust Region, and Deep Deterministic Policy Gradient. An average ensemble method was applied to obtain trading actions. We set SP100 component stocks as the portfolio pool and used 11 years of daily data to train the model and simulate the trading. Our method demonstrated better performance than the two benchmark methods and each individual algorithm without fuzzy extension. In practice, real market traders could use the trained model to make inferences and conduct trading, then retrain the model once in a while since training such models is time0consuming but making inferences is nearly simultaneous.

Keywords: stock portfolio management; reinforcement learning; deep learning; fuzzy set; ensemble

check for updates

Citation: Hao, Zheng, Haowei Zhang, and Yipu Zhang. 2023. Stock Portfolio Management by Using Fuzzy Ensemble Deep Reinforcement Learning Algorithm. *Journal of Risk and Financial Management* 16: 201. https://doi.org/10.3390/jrfm16030201

Academic Editor: David Liu

Received: 16 February 2023
Revised: 13 March 2023
Accepted: 13 March 2023
Published: 15 March 2023

1. Introduction

The goal of stock trading is to optimize some relevant measures of performance, such as profit and utility (Moody and Wu 1997). Traditionally, trading is performed by humans, who are classified as fundamentalists and technicalists (Murphy 1999), depending on the strategies one use. Stock trades based on algorithms have received attention and have started to take over the finance industry in the last two decades with the development of online trading platforms (Creamer and Freund 2010; Ozbayoglu et al. 2020). Since the successes that Deep Learning (DL) and Reinforcement Learning (RL) algorithms achieved in areas such as Go game (Silver et al. 2016) and video games (Vinyals et al. 2017), researchers have wanted to train computers to be skilled traders using these algorithms.

One of the ideas is supervised learning, for example, trading based on the prediction of market trends. Studies by (Oelschläger and Adam 2021) and (Leung et al. 2000) used statistical and DL algorithms to label the market as bullish and bearish; while (Sezer et al. 2017) provided a finer labeling by further giving each direction three levels. Another type of effort was given to the prediction of stock prices. Long short-term memory (LSTM) networks were used for the prediction of the Standard and Poor 500 (SP500) index in (Fischer and Krauss 2018). It, together with a number of other DL models, was implemented for the prediction of ten stock prices in (Balaji et al. 2018). In (Singh et al. 2023), the author implemented the Logistic model, Gompertz model, and Harvey model to predict the increasing trend of the adoption of Unified Payment Interface. However, maximizing profit by predicting market behavior or prices was a suboptimal approach since the prediction step itself suffered from forecast errors (Moody and Saffell 1998). Therefore, a number of past works have been performed to train the program to trade directly by optimizing some objective functions. The authors in (Moody et al. 1998) presented an adaptive algorithm called Recurrent Reinforcement Learning (RRL) to train the trading system for a single asset; (Gold 2003) investigated the foreign currency exchange market by using RRL; In (Zhang et al. 2020), the authors designed trading strategies by adopting Deep RL algorithms for future contracts and tested them on 50 future contracts individually. Although these approaches were able to obtain remarkable returns, their limitation was that the trading was only on one asset; that is, it was an allocation of money between a risk-free investment and only one risky investment.

Financial theories state that to achieve better investment return, one would need to diversify the investment or create a portfolio that usually has much smaller volatility or variance of returns compared to a single asset (Rubinstein 2002). In (Iliev et al. 2023), the authors provided a creative way of constructing the pool for building a portfolio. They focused on stocks that were best related to the concepts of Circular Economy and Environmental, Social, and Governmental principles and narrowed down to a top 100 stocks from a list of 6000 globally traded ones. The authors were able to compare three portfolios built from these 100 stocks to some benchmark measurements, such as a BlackRock ETF, and found that all portfolios outperformed them with larger returns and equal or smaller volatilities. In (Li and Hoi 2014), the author grouped related work into four categories depending on the theories or models that the strategies followed, including Follow-the-Winner, Follow-the-Loser, Pattern-Matching, and Meta-Learning. In (Ozbayoglu et al. 2020), the author also surveyed several studies regarding portfolio management. For instance, in (Fu et al. 2018), multiple stocks were analyzed and selected by using various DL algorithms; (Lee and Yoo 2020) proposed threshold-based portfolios that were built from a LSTM-based prediction model; (Liang et al. 2018) proposed an Adversarial Training method that was adopted from Policy Gradient algorithm and tested it on the Chinese stock market.

Jiang and Liang (2017) applied Convolutional Neural Networks (CNN) to the Deterministic Policy Gradient (DPG) algorithm to develop a portfolio of 12 cryptocurrencies and received remarkable return rates. In (Nikolaev and Petrova 2021), the author classified the stocks' weekly change into seven categories and implemented two models involving CNN to make predictions. Their results showed the effectiveness of CNN in predicting graphical representations such as stock price trends. Yang et al. (2020) used an ensemble strategy of three RL algorithms for a portfolio of 30 Dow Jones stocks. Our work was to build a model-free algorithm that can manage a stock portfolio and achieve high returns and small volatility without predicting stock prices. Sharpe ratio, a measurement introduced by Sharpe (1998), was computed by the expected return relative to its standard deviation. It is an effective measurement to evaluate portfolios with similar risk characteristics over the same period. For example, the authors in (Em et al. 2022) created some metrics for some and used the Sharpe ratio and its variants to evaluate performance differences among bond portfolios in the global market.

In sum, there is room to improve the existing literature. The originality of this proposed method includes the following: first, no prediction is needed, and attention is paid only to trading since predicting stocks' prices has errors and is different from making profits. Secondly, the method is able to include a much larger number of stocks to select from. We use SP100 index stocks, but the method could be scaled up easily to include more. Third, we want to include market trend information, such as bullish and bearish, quantitatively and continuously to help on decision-making. Last but not least, the method assumes no trading or financial knowledge. The trading agent (machine) will be learning by trial and error.

The motivation of this work is to use several RL algorithms to train a computer to manage a portfolio of a relatively large number of stocks and to receive higher measures of trading, such as profit or Sharpe ratio, by using only past data and assuming no other trading knowledge. To incorporate the market direction, e.g., bullish/bearish, but in a more quantitative way, we incorporated fuzzy extension. We also use average ensemble to lower the volatility and, therefore, gain higher Sharpe ratios. The remaining parts of this work are organized as follows. Section 2 reviewed RL algorithms and proposed our method; Section 3 presented the data sets and data preprocessing; Section 4 illustrated the results; and Section 5 summarized the work in conclusions and some possible future work.

2. Methodology

2.1. Model Terminology and Assumptions

We explain some notations and their meanings as well as assumptions needed for them in our problem.

- Reward, denoted by r, is a scalar feedback to reflect the performance of actions. The goal of most RL algorithms is to optimize the cumulative reward. For the stock trading problem, it depends on the current market and portfolio information and the trading actions taken, and, therefore, it is denoted as $r(\mathbf{s_{t-1}}, \mathbf{a_{t-1}}, \mathbf{s_t})$, meaning the change in the total portfolio value when facing state $\mathbf{s_{t-1}}$ and taking action $\mathbf{a_{t-1}}$, and, therefore, the arriving state $\mathbf{s_t}$, where $\mathbf{s_t}$ and $\mathbf{a_t}$ are defined later in this section.
- We consider a pool of M stocks over T time periods and assume a trader may buy or sell stocks but may not short sell. For each stock i, and time t, where $i = 1, 2 \ldots M$ and $t = 1, 2 \ldots T$, let p_t^i be the price and $n_t^i \geq 0$ be the numbers of share holding in the trader's portfolio. Respectively, combining p_t^i and n_t^i yields $\mathbf{p_t} = [p_t^1, p_t^2, \ldots, p_t^M]^T$ and $\mathbf{n_t} = [n_t^1, n_t^2, \ldots, n_t^M]^T$. Then, the portfolio's return at time t is

$$R_t = (\mathbf{p_t} - \mathbf{p_{t-1}})^T \mathbf{n_t} - cost \cdot \mathbf{p_{t-1}}^T |\mathbf{n_t} - \mathbf{n_{t-1}}| \tag{1}$$

  where *cost* is the transaction cost rate, usually ranging from 0% to 1%. Therefore, our objective is to maximize the total return, the sum of period-wise returns over all periods.

$$J = \sum_{t=1}^{T} R_t \tag{2}$$

- We define $\mathbf{s}$ as the environment state space vector, which will be the model input. In our problem, the environment state space includes two types of components: the quantities that are dependent on trading actions, e.g., account cash balance and stock shareholdings, which is denoted by vector $\mathbf{s_1}$; and the quantities that are independent of trading actions, e.g., stock prices and technical indicators, such as MACD, denoted by vector $\mathbf{s_2}$. Thus, $\mathbf{s} = [\mathbf{s_1}, \mathbf{s_2}]$.
- Action $\mathbf{a}$ is a vector with a dimension equaling the number of stocks, M, whose positive components indicate buying and negative components indicate selling of the corresponding stocks. Each component of action $\mathbf{a}$ is a number between -1 and 1, indicating the number of shares we buy or sell. For example, a number -0.1 would mean that we sell $0.1H$ shares of a stock, where H is a hyperparameter defining the maximum amount of shares for any single transaction. There are usually two ways to claim the maximum: the first way is to choose the same number of shares for all stocks; for example, $H = 1000$ would mean a single transaction can, at most, buy or sell 1000 shares of any stock; the second way is to choose the same value for all stocks, for example, a single transaction can at most have a total value of 100,000 USD. In this case, the number of maximum shares for each stock is different and is computed by 100,000 divided by the current stock price. In this work, we used the second way. $\pi(\mathbf{a}|\mathbf{s})$ is the stochastic policy that gives the probability of action $\mathbf{a}$ when facing state $\mathbf{s}$. We denote $\mathbf{a_t}$ as the action taken in period t. In our problem, for example, say for stock A, the action could be (1) buying k_1 shares of the stock and resulting in $n_t = n_{t-1} + k_1$ shares, or (2) holding, which results in $n_t = n_{t-1}$, or (3) selling k_2 stocks and yielding $n_t = n_{t-1} - k_2$. The action will also update the cash balance and other stocks and,

therefore, reach the state $\mathbf{s_t}$. This action results in a reward, denoted by $r(\mathbf{s_{t-1}}, \mathbf{a_{t-1}}, \mathbf{s_t})$, which is the change in the portfolio's total value.

- We also make the following assumptions and constraints. There will be no short selling. All selling transactions in the action vector are conducted before buying, and all buying transactions' total value does not exceed the current cash amount plus the total value of all sell transactions. That is, the traders could use the cash received from selling stocks in a period to buy other stocks in the same period, but they could not borrow other cash. In addition, every single transaction's total value does not exceed $K\%$ of the total portfolio value, where K is a hyperparameter.
- Value function $V^{\pi}(\mathbf{s})$ and Q function $Q^{\pi}(\mathbf{s}, \mathbf{a})$ are defined to be the value function of state $\mathbf{s}$ and the value function of state-action pair $(\mathbf{s}, \mathbf{a})$ by following policy π. In stock trading, $V^{\pi}(\mathbf{s})$ means the expected future gain when facing state $\mathbf{s}$, and $Q^{\pi}(\mathbf{s}, \mathbf{a})$ means the expected return when facing state $\mathbf{s}$ and taking action $\mathbf{a}$ if the trader follows policy π.
- Advantage function, denoted by $A^{\pi}(\mathbf{s}, \mathbf{a}) = Q^{\pi}(\mathbf{s}, \mathbf{a}) - V^{\pi}(\mathbf{s})$, can be thought of as how much better state $\mathbf{s}$ would be if we take action $\mathbf{a}$ opposed to average. In many algorithms, subtracting the value function is called a baseline. In our approach, value function, Q function, and advantage function are predefined and embedded in the programming packages and need no extra attention.

2.2. Reinforcement Learning Algorithms

A review of the classic Policy Gradient algorithm will be given by using the notations in the previous section, then followed by five RL algorithms, some of which are highly related to the classic Policy Gradient. These five algorithms are all suitable for continuous action and are used in our method.

In classic Policy Gradient, the objective of many RL learning algorithms is to maximize the expected value of the total reward $J = E[r(\tau)]$ where τ represents a trajectory of states and actions, and $r(\tau)$ is the total reward by taking this trajectory. Policy Gradient Theorem (Sutton and Barto 2018) provided a simplified form of the gradient of J that allowed the update of parameter θ by using gradient ascending, $\theta \leftarrow \theta + \eta \nabla_{\theta} J$. The theorem states that the gradient term is proportional to $E[\sum_t Q^{\pi}(s_t, a_t) \nabla_{\theta} \ln \pi_{\theta}(a_t|s_t)]$. One may subtract a base-line value function $V^{\pi}(s_t)$ from function $Q^{\pi}(s, a)$ to make negative values possible; then the Policy Gradient could be expressed as follows, which is then used in the gradient ascending method.

$$\nabla_{\theta} J = E[(Q^{\pi}(s, a) - V^{\pi}(s_t)) \nabla_{\theta} \ln \pi_{\theta}(a|s)] \tag{3}$$

(1) Advantage Actor-Critic (A2C)

A2C (Mnih et al. 2016) is an on-policy actor-critic algorithm. The method uses the advantage function to replace the return function with the advantage function $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$ in the Policy Gradient, which then becomes

$$\nabla_{\theta} J = E[\sum_{t=1}^{T} (Q(s_t, a_t) - V(s_t) \nabla \pi_{\theta}(a_t|s_t)]. \tag{4}$$

Next, $Q(s_t, a_t)$ can be approximated by the reward that is observed at time t, r_t plus the value at the next time, $V(s_{t+1})$. Thus, the gradient for parameter θ updating is

$$\nabla_{\theta} J = E[\sum_{t=1}^{T} (r_t + V(s_{t+1}) - V(s_t) \nabla \pi_{\theta}(a_t|s_t)]. \tag{5}$$

The algorithm uses one network to estimate the policy and another network to estimate the value function, which may share some parameters.

(2) Trust Region Policy Optimization (TRPO)

TRPO (Schulman et al. 2015) is an off-policy algorithm that uses an old policy, denoted as $\pi_{\theta'}$, to collect data for updating a new policy π_θ. The expected return object function (Sutton and Barto 2018) could be written in terms of both policies as

$$J(\theta) = E_t(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}\hat{A}(s_t, a_t)), \tag{6}$$

where $\hat{A}(s_t, a_t)$ is the estimated advantage function by using importance sampling (Kloek and Van Dijk 1978; Van Dijk and Kloek 1983), which allows approximating the expectation of the distribution of interest from samples generated from a different distribution. However, the expectation might be unstable if these two approximations are too far away; therefore, the two policies need to be somehow closed in the above objective function. TRPO uses a trust region defined by KL divergence (Kakade and Langford 2002) to restrict the choice of the new policy. In particular, the algorithm maximizes $J(\theta)$ among θ whose distance defined KL divergence to θ' is no larger than δ, or $D_{KL}(\theta, \theta') < \delta$, where δ is a small positive hyperparameter.

(3) Proximal Policy Optimization (PPO)

The idea of PPO (Schulman et al. 2017) started from the same objective function as in TRPO $J(\theta) = E_t(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}\hat{A}(s_t, a_t))$ and, therefore, would avoid have old and new policies too far away. Instead of using a trust region, PPO penalizes the policies distance by adding a clip function and using objected function

$$J^{clip}(\theta) = E_t[min\{\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}\hat{A}(s_t, a_t), clip(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)\hat{A}(s_t, a_t)\}] \tag{7}$$

where ϵ is a small positive number around 0.1–0.2, function $clip(a, b, c)$ returns a if $b < a < c$, returns b if $a < b$, and returns c if $a > c$. Therefore, it clips the ratio between new and old policies to be within $1-\epsilon$ and $1+\epsilon$. The method was proven to have better sample complexity and be simpler to implement than some other online Policy Gradient methods and provides a favorable balance between sample complexity, simplicity, and wall time (Achiami et al. 2016; Schulman et al. 2017).

(4) Actor-Critic Using Kronecker Factored Trust Region (ACKTR)

ACKTR (Wu et al. 2017) is a natural Policy Gradient method with trust region optimization and uses a more accurate second-order optimization. Similar to A2C, ACKTR uses two networks for policy and another network for the advantage function. On some classical testing tasks, the algorithm obtained two to three-fold sample efficiency and only had 10–25% more computation cost due to the Kronecker-factored approximated curvature (K-FAC) (Martens and Grosse 2015).

The algorithm uses $\theta \to \theta + \eta F^{-1}\nabla_\theta J$ for updating, where η is the learning rate, F is the Fisher information matrix that will be approximated by K-FAC (Martens and Grosse 2015), and $\nabla_\theta J$ is the Policy Gradient. The learning rate η in ACKTR involves trust region radius hyperparameter δ and is given by $\eta = \sqrt{\frac{2\eta}{(\nabla_\theta J)^T F^{-1} \nabla_\theta J}}$.

(5) Deep Deterministic Policy Gradient (DDPG)

DDPG (Lillicrap et al. 2015) uses deep neural networks for fitting a policy $\pi(s_t)$ called the actor with actor parameters θ^π and critic Q function, $Q(S_t, a_t)$ with critic parameters θ^Q. It adapts Deep Q-Learning (DQN) to continuous action cases. The method is an off-policy algorithm using a replay buffer that stores (state, action, reward, next state) or (s_t, a_t, r_t, s_{t+1}) to train networks and has a deterministic actor policy function $\pi(s_t)$.

To update parameter θ^Q, DDPG uses the recursive relationship from the Bellman equation

$$Q(s_t, a_t) = E[r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})] \tag{8}$$

where γ is a discount factor whose value is usually between 0.9 and 0.99 and minimizes the mean square error between $Q(s_t, a_t)$ and $r(s_t, a_t) + Q(s_{t+1}, \pi(s_{t+1}))$, or

$$L(\theta^Q) = E[(r(s_t, a_t) + Q(s_{t+1}, \pi(s_{t+1})) - Q(s_t, a_t))^2] \tag{9}$$

Then, the method updates the parameters of the actor, using the gradient of expected return with respect to θ^π

$$\nabla_{\theta^\pi} = E_t[\nabla_a Q(s_t, a_t|\theta^Q)\nabla_{\theta^\pi}\pi(s_t|\theta^\pi)] \tag{10}$$

2.3. Fuzzy Extension

It has been known that trading strategies work differently in a bullish market and in a bearish market (Chen et al. 2020; Dai et al. 2012). Effort has been made for bullish/bearish market classification(Chong et al. 2017; Oelschläger and Adam 2021; Wu et al. 2020). However, many classification methods could only tell until a turning signal was shown, and they only qualitatively classified the market. The idea of a fuzzy set state is where the state of the membership function, whose range is between 0 and 1, measures the grade that an element belongs to a set (Pal and Bezdek 1994). Our approach, instead of classifying each trading day as bullish, bearish, or oscillation, wanted to quantify, for example, a trading day to be 0.7 oscillation degree, 0.2 bullish degree, and 0.1 bearish degree. To achieve this, we adopted the approach in (Deng et al. 2016; Lin et al. 2006), as follows:

For each stock, first, find the rate of return for each of the time periods by $r_t = \frac{p_t - p_{t-1}}{p_{t-1}}$. To obtain its degree for $k = 3$ fuzzy sets representing bullish, bearish, and oscillation markets for a time period t:

- choose moving window length T_0, and using k-mean clustering on the rates of return $r_t, r_{t-1}, r_{t-2}, \ldots, r_{t-T_0+1}$ to get $k = 3$ groups with means and standard deviations, m_j and s_j, $j = 1, 2, 3$, respectively.
- For period t_0, compute fuzzy degree f^j, $j = 1, 2, 3$ via the Gaussian membership function (Krasnyuk et al. 2022; Lin et al. 2006) as

$$f^j = e^{-(r_t - m_j)^2/s_j^2} \tag{11}$$

- These fuzzy degrees, f_t^1, f_t^2, f_t^3, are added as fuzzy extensions to obtain the final state space, which was stated in Section 3.
- repeat this process for all stocks and for all periods.

2.4. Ensemble

We use the RL algorithms, including A2C, PPO, DDPG, ACKTR, and TRPO, which are all suitable for continuous action to train the agent individually. These algorithms each have their strengths and weaknesses, and even for stock trading, they may behave well in one period but bad in another (Yang et al. 2020), which we observe in our initial trials as well. We train the model individually and take the average over the five output actions as our final action decision. Concretely, for any time period t, the model implements each of the five algorithms mentioned above on a training data set and obtains five action vector outputs, say $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$, $\mathbf{a_4}$, and $\mathbf{a_5}$, with each of them representing the decision for buying/holding/selling each of the stocks and the shares of the transaction as defined in the Model Terminology section. Then, the ensemble technic averages the five actions and yields action $\bar{\mathbf{a}} = \frac{\mathbf{a_1}+\mathbf{a_2}+\mathbf{a_3}+\mathbf{a_4}+\mathbf{a_5}}{5}$ as the decision for trading in period t. It is worth noting that since each of the five $\mathbf{a_i}$ satisfies the trading constraints claimed in the Model Terminology section, their average action $\bar{\mathbf{a}}$ will also satisfy them.

A complete diagram of our work-frame is shown in Figure 1. First, environment information is observed, and fuzzy extension is computed and added to feed the model as input for trading period t. Second, five RL algorithms are trained and provide five actions, which are then averaged to receive the final action decision. Next, this action is conducted to yield a reward and update part of the environment's information, such as shareholdings and cash. Finally, the updated environment information, together with the reward, is used for period $t + 1$, and the whole process repeats until all time periods are used.

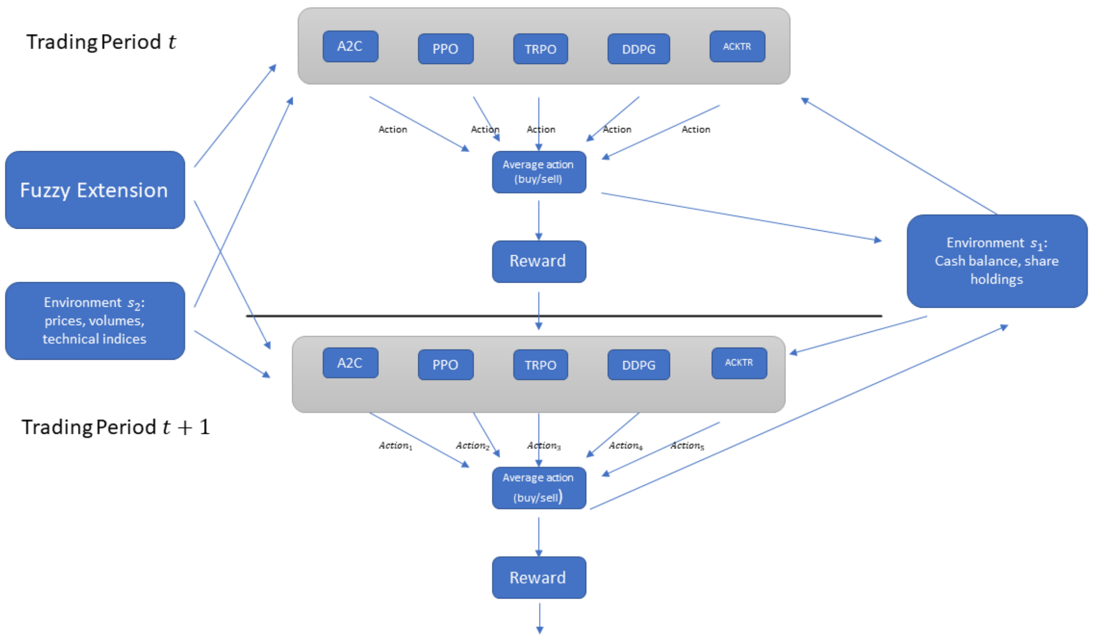

Figure 1. Model Flow Chart.

3. Datasets and Preprocessing

We considered all component stocks in the S&P 100 index during the period of 2011–2021, except for those that were listed on the market later than 2011. The data were available at the daily level from yahoo finance, which yielded $M = 94$ stocks that made up our portfolio pool with $T = 2537$ trading days. For each trading day, t, we used $M = 96$ dimension vectors to denote the shareholdings $\mathbf{n_t}$, the daily open price $\mathbf{o_t}$, high price $\mathbf{h_t}$, low price $\mathbf{l_t}$, adjusted closed prices $\mathbf{p_t}$, and volumes $\mathbf{v_t}$. Following the steps in Section 2.3 for the fuzzy extensions, we chose a moving window length of $T_0 = 60$, and obtained three 96-d vectors, $\mathbf{f_t^1}, \mathbf{f_t^2}, \mathbf{f_t^3}$. A snapshot of the computation for the fuzzy extensions on one sample trading day is given in Figure 2 and Table 1.

Table 1. The fourth column shows the fuzzy extension of AAPL on 8 April 2019, which was based on a daily return of 0.0115, or 1.15%.

	Cluster Mean	Cluster SD	Fuzzy Extension Value
increasing	0.0411	0.0133	$e^{-(0.0115-0.0411)^2/0.0133^2} = 0.0083$
oscillation	0.0029	0.01	$e^{-(0.0115-0.0029)^2/0.01^2} = 0.4369$
decreasing	-0.0323	0.0108	$e^{-(0.0115+0.0323)^2/0.0108^2} = 4.9 \times 10^{-8}$

There were a larger number of technical indicators to measure stock movement in various ways, among which we chose the ones that were considered related to stock trends and buy/sell signals (Di 2014; Vargas et al. 2018), including Moving Average Convergence Divergence (MACD) $\mathbf{MACD_t}$, Williams Overbought/Oversold index (WR) $\mathbf{WR_t}$, Relative Strength Index (RSI) $\mathbf{RSI_t}$, Commodity Channel Index (CCI) $\mathbf{CCI_t}$, and Average Directional Index (ADX) $\mathbf{ADX_t}$. Combining these indicators into one vector yielded $\mathbf{TI_t}$. The daily account cash balance is denoted by c_t. Following the notations in Section 2.1, we have $\mathbf{s_1} = [c_t, \mathbf{n_t}]$, and $\mathbf{s_2} = [\mathbf{o_t}, \mathbf{h_t}, \mathbf{l_t}, \mathbf{p_t}, \mathbf{v_t}, \mathbf{TI_t}]$.

Finally, we obtained the state space for our RL algorithms as $\mathbf{s} = [\mathbf{f}^1_t, \mathbf{f}^2_t, \mathbf{f}^3_t, \mathbf{s}_1, \mathbf{s}_2] = [\mathbf{f}^1_t, \mathbf{f}^2_t, \mathbf{f}^3_t, c_t, \mathbf{n}_t, \mathbf{o}_t, \mathbf{h}_t, \mathbf{l}_t, \mathbf{p}_t, \mathbf{v}_t, \mathbf{TI}_t]$, which has dimensions of $14 \times 94 + 1 = 1317$.

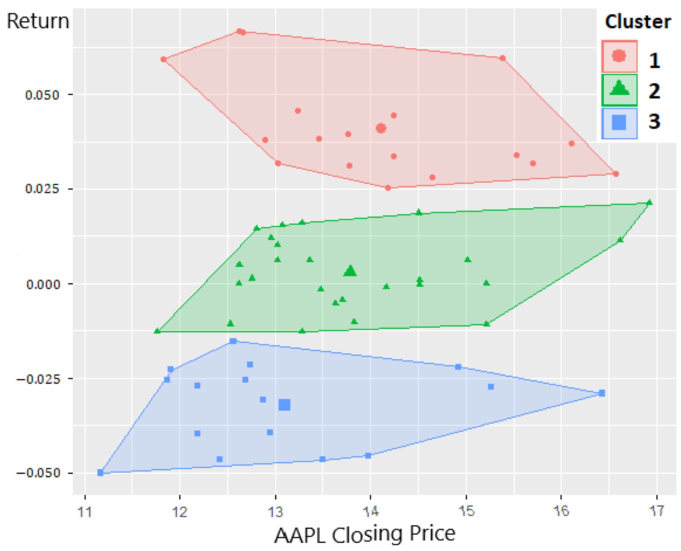

Figure 2. A snapshot of three-means clustering by using AAPL daily returns from 12 January 2019 to 7 April 2019. The plot included the closing prices as the horizontal axis for a better view, although the clustering was performed only using the return, which was the vertical axis.

4. Results

For the first round of training, the dataset was split into a training interval from the beginning of 2011 to the end of 2019, a testing interval ranging from 18 December 2020 to 20 January 2021 (20 trading days), and a validation interval containing any trading day in between, which had about 250 days. In each of the following rounds, training, testing, and validation intervals were all moved forward by 20 trading days. The process repeated until the last day of the data was used. Notice the reason that we chose 18 December 2020 as the first testing trading day was to ensure that suing periods of 20 trading days, the last trading day could exactly be 31 December 2021. A diagram showing the training/validation/testing periods update is shown in Figure 3.

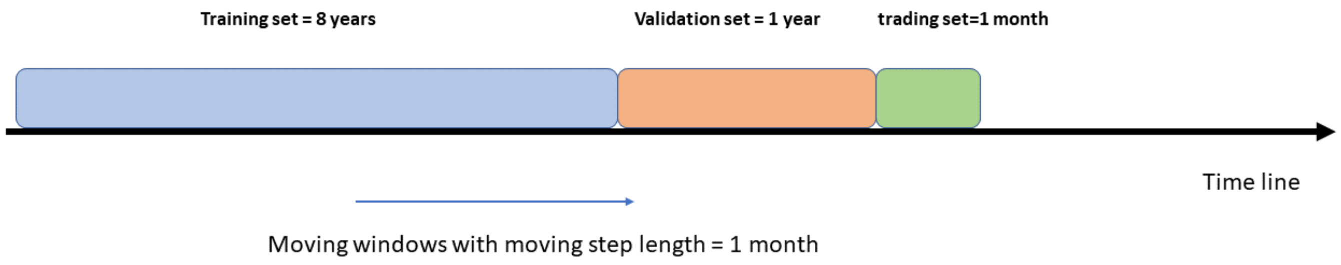

Figure 3. Data division and rolling (we used 20 trading days as 1 month).

We used a hypothetical initial cash amount of 10 million US dollars and conducted the trial 100 times. The histograms for final return rates and Sharpe ratios were given in Figures 4 and 5. The red vertical line in Figure 4 indicated a SP100 rate of return during the same period. We observed that most of the trials were able to gain higher rates of return than the index. In Figure 5, the red vertical line marked the SP100 index Sharpe ratio, which was about the same as the average Sharpe ratio for our 100 trials.

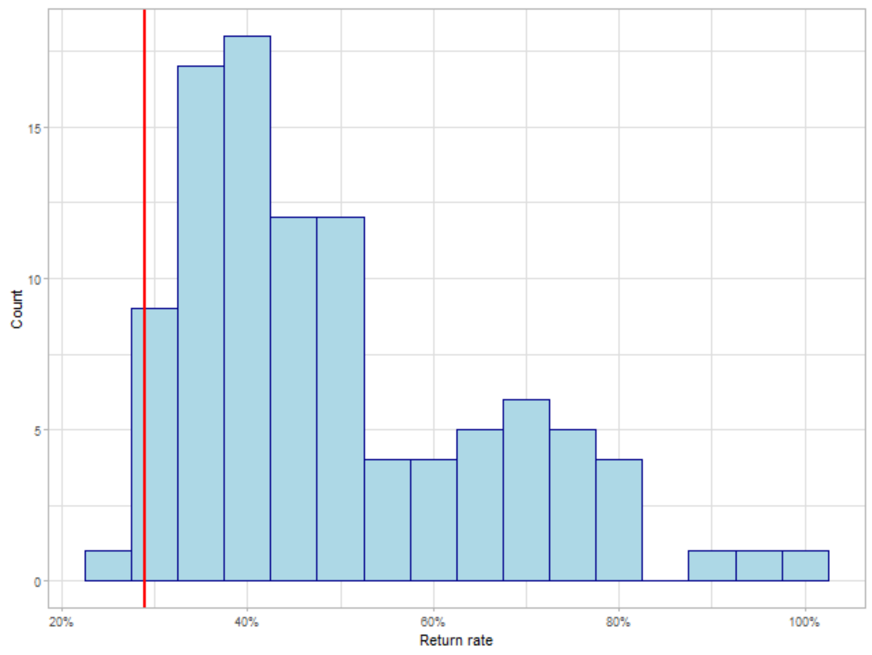

Figure 4. Histogram for return rates of 100 trials by using our fuzzy ensemble model. The red vertical line indicates the SP100 index return rate.

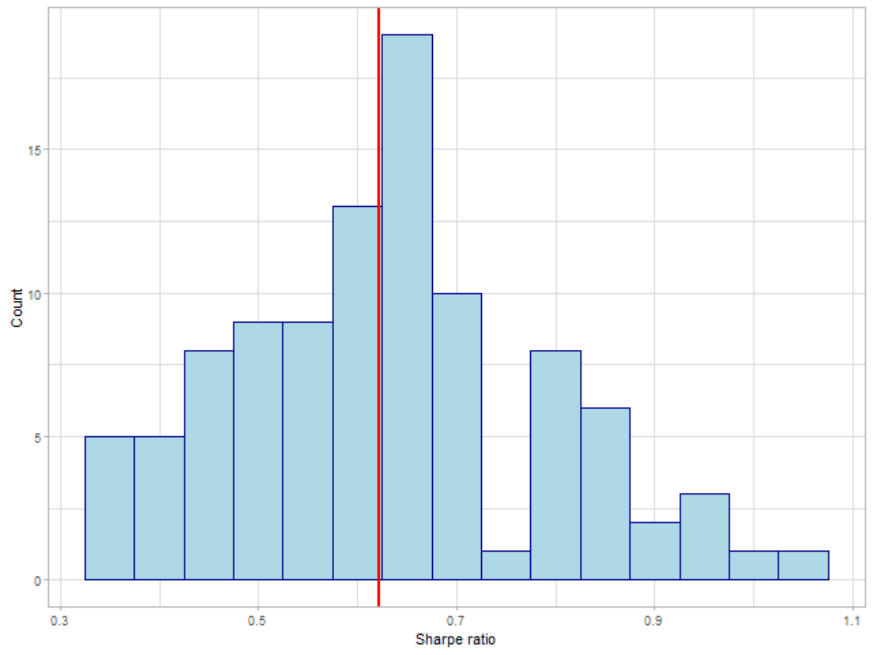

Figure 5. Histogram for Sharpe ratio of 100 trials by using our fuzzy ensemble model. The red vertical line indicates the SP100 index Sharpe ratio.

We took the average portfolio values over the 100 trials at the end of each trading period. A time series plot of our results is shown in Figure 6, as well as two benchmark portfolios with SP100 representing buying and holding the S&P100 index and rebalance (Davis and Nairn 2012) being the strategy that money was evenly spread over all stocks and the weights were rebalanced to maintain the even value allocation at the end of each trading interval. One could see that the average final portfolio value was over 14.9 million or a return rate of 49%. A 95% simultaneous confidence band for the portfolio value using Bonferroni's adjustment (Dunn 1961) was given by the shaded region. The final value for the S&P100 index was 12.9 million or a return rate of 29%; the rebalancing strategy yielded a return rate of 29.7%. We could see that our method's confidence band was able to stay above the S&P100 index and rebalancing for the whole period. Thus, our method was able to achieve a higher return while keeping about the same Sharpe ratio as the market index.

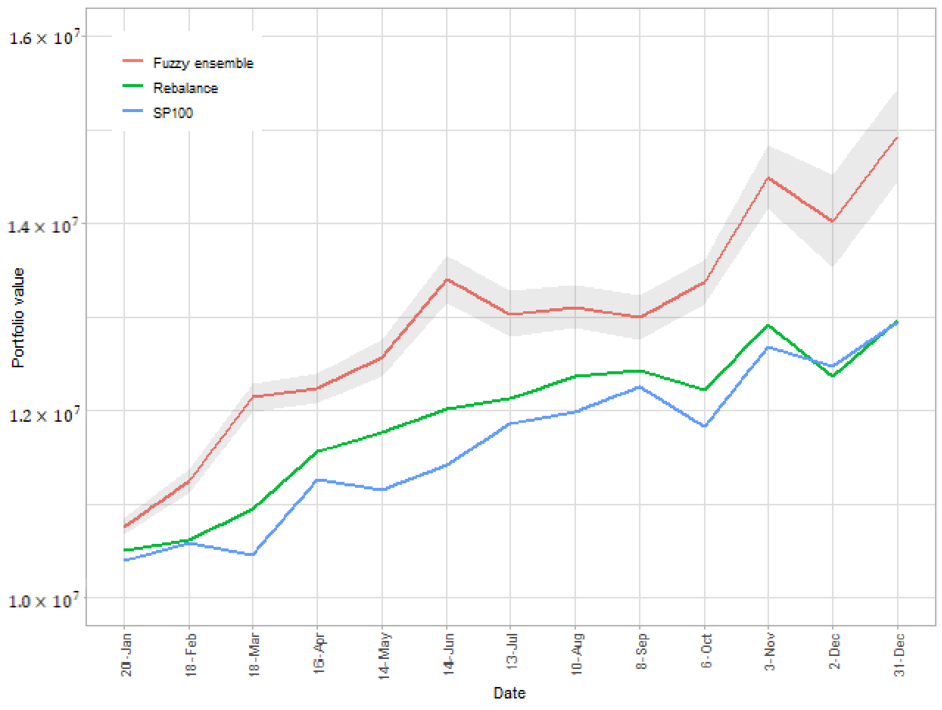

Figure 6. Time series plot with a 95% confidence band for the portfolio value of our fuzzy ensemble model compared to the SP100 index and rebalance portfolio with a starting amount of 10,000,000 USD.

For comparison purposes, we respectively used each of the five classic RL methods without a fuzzy extension to conduct the experiment 100 times and obtained their mean portfolio values. The result is given in Figure 7. One can see that only the ACKTR model was able to have a final portfolio value a bit higher than our method, and it was still lower than the upper limit of the 95% confidence band. However, all methods, including ACKTR, showed more volatility, especially during the last three periods when the market had large oscillations. More quantitative results are summarized in Table 2. One could see that our method had a Sharpe ratio of 0.627, which was close to the S&P100 index Sharpe ratio of 0.622 and rebalancing portfolio Sharpe ratio of 0.679, but had a much higher return rate. None of the five RL methods had a Sharpe ratio of more than 0.57, with the lowest Sharpe ratio of 0.347 belonging to DDPG, which indicated that to reach similar expected return levels, one must suffer from larger volatilities.

Table 2. The results of total returns and Sharpe ratios.

	Ours	**SP100**	**Rebalance**	**A2C**	**ACKTR**	**DDPG**	**TRPO**	**PPO**
Return Rate	49.4%	29.4%	29.9%	38.2%	44.8%	27.9%	35.9%	42.3%
Sharpe Ratio	0.627	0.622	0.679	0.452	0.565	0.347	0.478	0.563

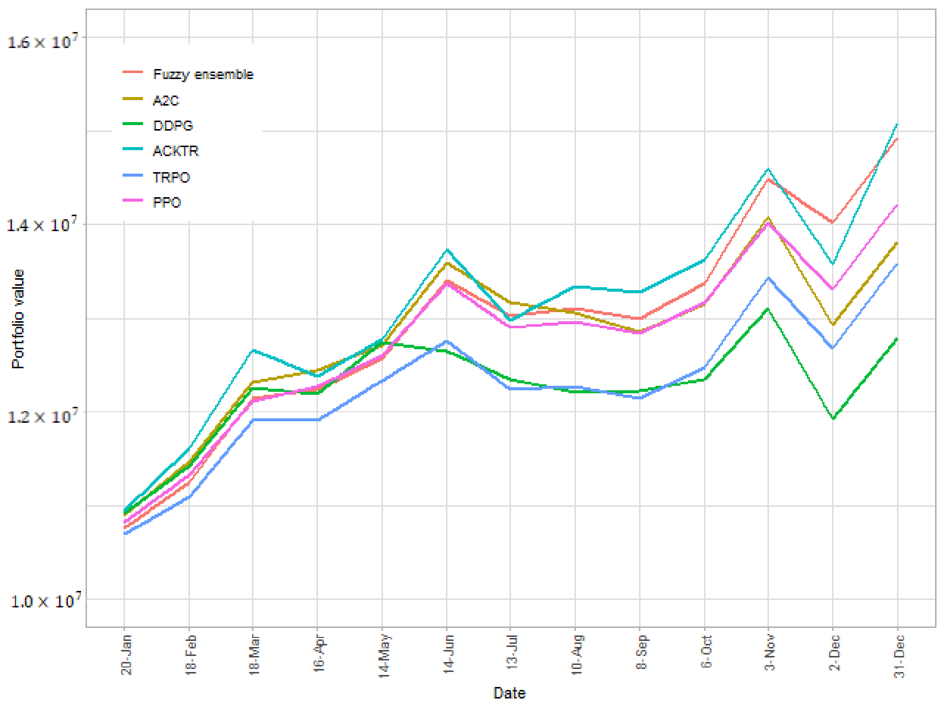

Figure 7. Time series plot for the portfolio value of our fuzzy ensemble model compared to other RL algorithms with a starting amount of 10,000,000 USD.

5. Discussion and Conclusions

In this work, we proposed and trained an agent that could conduct automated stock trading from a pool of stocks to form a portfolio and beat the market. The work had two theoretical contributions.

First, the method was able to handle a vast number of stocks and gain a satisfactory return with no more risks. In our case, the model built and managed a portfolio from S&P 100 component stocks. The model could easily include an even larger number of stocks. In the existing literature, the difficulties of dealing with many of stocks came from two aspects. The first one was the restriction of the model. For example, time series type analysis or Markov Decision Process were naturally limited to small dimension state spaces. The other one is the computing capacity of the computer. In our approach, the most time-consuming part was training, but this training and retraining could be performed during closed market periods daily or weekly. The trading operations were nearly simultaneous after the model was trained. The model earned an average return of about 49% during a one-year period. This was significantly higher than the return of the index and rebalanced equal-weight portfolio, which was about 29%. Their associated Sharpe ratios were similar and around 0.6.

Second, the method incorporated ensemble technic and fuzzy extension in addition to existing RL algorithms. The averaging ensemble technic reduced the volatility compared to any single algorithm, and the fuzzy extension itself also provided a numerical way to describe the market trend as opposed to the traditional categorical classifications as bullish/bearish/oscillation. As the comparison results suggested, our method achieved higher returns and Sharpe ratios than each single RL algorithm without using the fuzzy extension.

Our work added evidence that machines have the potential to beat humans and provided investors with a practical way to make profit in the market. Most investors, especially individual investors and small funds used fundamental analysis and technical analysis techniques for trading. It was not easy for them to beat the indices consistently, such as S&P 100, S&P 500, Dow Jones Industrial Average, and NASDAQ Composite. The reasons were that for fundamental analysis, most information they collected was public information and, therefore, was already reflected in current prices (price-in); while for technical analysis, the patterns or technical indicators might be used or interpreted in more than one way, which might contradict others. Our method assumed no fundamental analysis, technical analysis, or other trading strategies, creating a possibility for them to achieve higher returns without running into more risks. The approach could also be used by large investment companies that have more and quicker information and data on a larger scale. It was desirable that the model could be trained to achieve even higher performance.

This work took a step into algorithm-based trading; however, it has limitations, which were from the nature of the market assumptions we used. We assumed that in the market: (1) Each of our transaction amounts was insignificant compared to the total volumes, thus had no impact on the market prices, and (2) we were able to sell or buy stocks right at the stocks' closing prices and the transactions were executed immediately.

The limitation of the first assumption was the scaling problem. When the portfolio value became large, this was no longer true. While it was somehow realistic as long as the total portfolio values were small, which in this work was 10 million US dollars, or the stocks volumes and market caps were reasonably large, which all S&P 100 stocks should satisfy. The limitation of the second assumption was that trading at the closing prices was not always possible, and only using daily closing prices would ignore the intraday trends and extended-hours trends and, therefore, miss many profitable opportunities.

There could be three possible ways for future study. The first way is based on the second assumption mentioned above. For example, if one wants to use computer algorithms for real automated trading, it would be necessary to use market information on finer time intervals, such as hourly data or minute data. Secondly, we could include training periods that include some recent long bullish and long bearish markets and tune the parameters so the model could be more robust. Historically speaking, U.S. stock markets had much longer bullish periods than bearish periods, which was also true in our data time frame. We are observing a long ongoing bearish market starting year 2022. Thus the authors think this period would be valuable for any algorithm-based trading strategy and would like to include it in the future. The third one is to improve the algorithms themselves. This includes adopting existing ones, such as implementing CNN to pre-transform the inputs, integrating additional features, such as ESG and macroeconomics indicators, into the model environment, and exploring more complicated structures or models.

Author Contributions: Conceptualization, Y.Z.; methodology, H.Z. and Z.H.; software, H.Z.; validation, Z.H. and Y.Z.; formal analysis, Z.H.; investigation, Z.H.; data curation, Z.H. and H.Z.; writing—original draft preparation, Z.H.; writing—review and editing, Z.H., H.Z. and Y.Z.; supervision, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All data in this study were obtained from yahoo finance.

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

Abbreviations

The following abbreviations are used in this manuscript:

DL	Deep Learning
RL	Reinforcement Learning
LSTM	Long short-term memory
RRL	Recurrent Reinforcement Learning
CNN	Convolutional Neural Network
MACD	Moving Average Convergence/Divergence
A2C	Advantage Actor-Critic
TRPO	Trust Region Policy Optimization
PPO	Proximal Policy Optimization
ACKTR	Actor-Critic Using Kronecker Factored Trust Region
K-FAC	Kronecker-factored approximated curvature
DDPG	Deep Deterministic Policy Gradient

WR	Williams Overbought/Oversold Index
RSI	Relative Strength Index
CCI	Commodity Channel Index
ADX	Average Directional Index

References

Achiam, Joshua, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. *Proceedings of the 34th International Conference on Machine Learning* PMLR 70: 22–31.

Balaji, A. Jayanth, D. S. Harish Ram, and Binoy B. Nair. 2018. Applicability of deep learning models for stock price forecasting an empirical study on bankex data. *Procedia Computer Science* 143: 947–53. [CrossRef]

Chen, Peng, Dongyun Yi, and Chengli Zhao. 2020. Trading strategy for market situation estimation based on hidden markov model. *Mathematics* 8: 1126. [CrossRef]

Chong, Eunsuk, Chulwoo Han, and Frank C. Park. 2017. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications* 83: 187–205. [CrossRef]

Creamer, Germán, and Yoav Freund. 2010. Automated trading with boosting and expert weighting. *Quantitative Finance* 10: 401–20. [CrossRef]

Dai, Min, Hefei Wang, and Zhou Yang. 2012. Leverage management in a bull–bear switching market. *Journal of Economic Dynamics and Control* 36: 1585–99. [CrossRef]

Davis, Jonathan, and Alasdair Nairn. 2012. *Templeton's Way with Money*. New York: Wiley Online Library.

Deng, Yue, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems* 28: 653–64. [CrossRef]

Di, Xinjie. 2014. *Stock Trend Prediction with Technical Indicators Using SVM*. Independent Work Report. Standford: Leland Stanford Junior University, USA.

Dunn, Olive Jean. 1961. Multiple comparisons among means. *Journal of the American Statistical Association* 56: 52–64. [CrossRef]

Em, Olga, Georgi Georgiev, Sergey Radukanov, and Mariana Petrova. 2022. Assessing the market risk on the government debt of kazakhstan and bulgaria in conditions of turbulence. *Risks* 10: 93. [CrossRef]

Fischer, Thomas, and Christopher Krauss. 2018. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research* 270: 654–69. [CrossRef]

Fu, Xingyu, Jinhong Du, Yifeng Guo, Mingwen Liu, Tao Dong, and Xiuwen Duan. 2018. A machine learning framework for stock selection. *arXiv* arXiv:1806.01743.

Gold, Carl. 2003. FX trading via recurrent reinforcement learning. Paper presented at 2003 IEEE International Conference on Computational Intelligence for Financial Engineering, Hong Kong, China, March 20–23; pp. 363–70.

Iliev, Nikola, Marin Marinov, Valentin Milinov, and Mariana Petrova. 2023. Is investment portfolio construction sustainable in the circular economy paradigm—The case of esg investment? In *Circular Business Management in Sustainability. ISCMEE 2022*. Lecture Notes in Management and Industrial Engineering; Cham: Springer, pp. 15–42.

Jiang, Zhengyao, and Jinjun Liang. 2017. Cryptocurrency portfolio management with deep reinforcement learning. Paper presented at 2017 Intelligent Systems Conference (IntelliSys), London, UK, September 7–8; Piscataway: IEEE, pp. 905–13.

Kakade, Sham, and John Langford. 2002. Approximately optimal approximate reinforcement learning. Paper presented at the Nineteenth International Conference on Machine Learning, San Francisco, CA, USA, July 8–12; pp. 267–74.

Kloek, Teun, and Herman K. Van Dijk. 1978. Bayesian estimates of equation system parameters: an application of integration by monte carlo. *Econometrica: Journal of the Econometric Society* 46: 1–19. [CrossRef]

Krasnyuk, Maxim, Iryna Hrashchenko, Svitlana Goncharenko, and Svitlana Krasniuk. 2022. Hybrid application of decision trees, fuzzy logic and production rules for supporting investment decision making (on the example of an oil and gas producing company). *Access Journal* 3: 278–91. [CrossRef] [PubMed]

Lee, Sang, II, and Seong Joon Yoo. 2020. Threshold-based portfolio: the role of the threshold and its applications. *The Journal of Supercomputing* 76: 8040–57. [CrossRef]

Leung, Mark T., Hazem Daouk, and An-Sing Chen. 2000. Forecasting stock indices: A comparison of classification and level estimation models. *International Journal of Forecasting* 16: 173–90. [CrossRef]

Li, Bin, and Steven CH Hoi. 2014. Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)* 46: 1–36. [CrossRef]

Liang, Zhipeng, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li. 2018. Adversarial deep reinforcement learning in portfolio management. *arXiv* arXiv:1808.09940.

Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv* arXiv:1509.02971.

Lin, Chin-Teng, Chang-Mao Yeh, Sheng-Fu Liang, Jen-Feng Chung, and Nimit Kumar. 2006. Support-vector-based fuzzy neural network for pattern classification. *IEEE Transactions on Fuzzy Systems* 14: 31–41.

Martens, James, and Roger Grosse. 2015. Optimizing neural networks with kronecker-factored approximate curvature. Paper presented at 32nd International Conference on Machine Learning, Lille, France, July 6–11; New York: PMLR, pp. 2408–17.

Mnih, Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. Paper presented at 33rd International Conference on Machine Learning, New York, NY, USA, June 19–24; New York: PMLR, pp. 1928–37.

Moody, John, and Lizhong Wu. 1997. Optimization of trading systems and portfolios. Paper presented at IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr), New York, NY, USA, March 24–25; Piscataway: IEEE, pp. 300–7.

Moody, John, and Matthew Saffell. 1998. Reinforcement learning for trading. *Advances in Neural Information Processing Systems*, 917–23.

Moody, John, Lizhong Wu, Yuansong Liao, and Matthew Saffell. 1998. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting* 17: 441–70. [CrossRef]

Murphy, John J. 1999. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. Penguin; New York: New York Institute of Finance.

Nikolaev, Daniel, and Mariana Petrova. 2021. Application of simple convolutional neural networks in equity price estimation. Paper presented at 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), Kharkiv, Ukraine, October 5–7; Piscataway: IEEE, pp. 147–50.

Oelschläger, Lennart, and Timo Adam. 2021. Detecting bearish and bullish markets in financial time series using hierarchical hidden markov models. *arXiv* arXiv:2007.14874 [CrossRef]

Ozbayoglu, Ahmet Murat, Mehmet Ugur Gudelek, and Omer Berat Sezer. 2020. Deep learning for financial applications: A survey. *Applied Soft Computing* 93: 106384. [CrossRef]

Pal, Nikhil R., and James C. Bezdek. 1994. Measuring fuzzy uncertainty. *IEEE Transactions on Fuzzy Systems* 2: 107–18. [CrossRef] [PubMed]

Rubinstein, Mark. 2002. Markowitz's "portfolio selection": A fifty-year retrospective. *The Journal of Finance* 57: 1041–45. [CrossRef]

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv* arXiv:1707.06347.

Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. Paper presented at 32nd International Conference on Machine Learning, Lille, France, July 6–11; New York: PMLR, pp. 1889–97.

Sezer, Omer Berat, Murat Ozbayoglu, and Erdogan Dogdu. 2017. A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters. *Procedia Computer Science* 114: 473–80. [CrossRef]

Sharpe, William F. 1998. The sharpe ratio. *Streetwise–the Best of the Journal of Portfolio Management* 3: 169–85. [CrossRef]

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529: 484–89. [CrossRef]

Singh, Sanjay Kumar, Shivendra Sanjay Singh, and Vijay Lakshmi Singh. 2023. Predicting adoption of next generation digital technology utilizing the adoption-diffusion model fit: The case of mobile payments interface in an emerging economy. *Access Journal* 4: 130–48. [CrossRef]

Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. Cambridge: MIT press.

Van Dijk, Herman K., and Teunis Kloek. 1983. *Experiments with Some Alternatives for Simple Importance Sampling in Monte Carlo Integration*. Technical report. Amsterdam: Elsevier.

Vargas, Manuel R., Carlos E. M. Dos Anjos, Gustavo L. G. Bichara, and Alexandre G. Evsukoff. 2018. Deep leaming for stock market prediction using technical indicators and financial news articles. Paper presented at 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, July 8–13; Piscataway: IEEE, pp. 1–8.

Vinyals, Oriol, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, and et al. 2017. A new challenge for reinforcement learning. *arXiv* arXiv:1708.04782.

Wu, Dingming, Xiaolong Wang, Jingyong Su, Buzhou Tang, and Shaocong Wu. 2020. A labeling method for financial time series prediction based on trends. *Entropy* 22: 1162. [CrossRef] [PubMed]

Wu, Yuhuai, Elman Mansimov, Roger B. Grosse, Shun Liao, and Jimmy Ba. 2017. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in Neural Information Processing Systems*; Cambridge: MIT Press, pp. 5279–88.

Yang, Hongyang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. 2020. Deep reinforcement learning for automated stock trading: An ensemble strategy. Paper presented at the first ACM International Conference on AI in Finance, New York, NY, USA, October 15–16; pp. 1–8.

Zhang, Zihao, Stefan Zohren, and Stephen Roberts. 2020. Deep reinforcement learning for trading. *The Journal of Financial Data Science* 2: 25–40. [CrossRef]