electronics

MDPI

Article

A Novel Anti-Risk Method for Portfolio Trading Using Deep Reinforcement Learning

Han Yue, Jiapeng Liu *, Dongmei Tian and Qin Zhang

College of Economics and Management, China Jiliang University, Hangzhou 310018, China; s20071201039@cjlu.edu.cn (H.Y.); z1907120102@cjlu.edu.cn (D.T.); s20071201042@cjlu.edu.cn (Q.Z.)
* Correspondence: jpliu@cjlu.edu.cn

Abstract: In the past decade, the application of deep reinforcement learning (DRL) in portfolio management has attracted extensive attention. However, most classical RL algorithms do not consider the exogenous and noise of financial time series data, which may lead to treacherous trading decisions. To address this issue, we propose a novel anti-risk portfolio trading method based on deep reinforcement learning (DRL). It consists of a stacked sparse denoising autoencoder (SSDAE) network and an actor–critic based reinforcement learning (RL) agent. SSDAE will carry out off-line training first, while the decoder will used for on-line feature extraction in each state. The SSDAE network is used for the noise resistance training of financial data. The actor–critic algorithm we use is advantage actor–critic (A2C) and consists of two networks: the actor network learns and implements an investment policy, which is then evaluated by the critic network to determine the best action plan by continuously redistributing various portfolio assets, taking Sharp ratio as the optimization function. Through extensive experiments, the results show that our proposed method is effective and superior to the Dow Jones Industrial Average index (DJIA), several variants of our proposed method, and a state-of-the-art (SOTA) method.

Keywords: portfolio trading; deep reinforcement learning; deep learning; stacked autoencoder; advantage actor–critic

Citation: Yue, H.; Liu, J.; Tian, D.; Zhang, Q. A Novel Anti-Risk Method for Portfolio Trading Using Deep Reinforcement Learning. *Electronics* **2022**, *11*, 1506. https://doi.org/10.3390/electronics11091506

Academic Editors: Andrea Prati, Luis Javier García Villalba and Vincent A. Cicirello

Received: 18 April 2022
Accepted: 5 May 2022
Published: 7 May 2022

1. Introduction

Portfolio trading aims to allocate resources in a way that maximizes returns while minimizing risk. Many prior studies have employed reinforcement learning (RL) algorithms to address a variety of stock trading problems and have proved both the potential and promising results of deep reinforcement learning (DRL) in financial markets [1–5].

RL combines the tasks of "forecasting" and "portfolio construction" into one integration step so as to closely combine the machine learning problem with the objectives of investors. The process that traders use to invest in stocks to maximize profits is the same as how the RL agents interact with the environment to obtain rewards and maximize the cumulative reward. RL can also easily consider important constraints such as transaction cost [6], market liquidity, and investors' risk aversion.

However, the existing methods have some common limitations. When confronted with a complex market environment, agents frequently struggle to find the optimal strategy. Li [7] has proved that using feature preprocessing in a trading algorithm can obtain higher transaction performance. More crucially, the stock market will be influenced by political, economic, and natural disasters (such as COVID-19); in other words, the stock market has externalities, and the impact of external shocks will cause the stock price to rise or fall abnormally.

In this paper, we employed the autoencoder (AE) network in the RL framework. AE has been used for noise resistance training in many fields to improve robustness. Qi et al. [8] proposed a robust stacked autoencoder (R-SAE) based on maximal correntropy criteria

(MCC). The proposed method outperforms other machine learning methods on the MNIST benchmark dataset in noise-proof ability. Li et al. [9] introduced a deep sparse autoencoder with lossless and nonnegative constraints, which increases the model's robustness and the performance of anti-interference. Therefore, we propose a novel anti-risk portfolio trading system based on DRL. The trading system consists of a stacked sparse denoising autoencoder (SSDAE) and an actor–critic based RL agent. The noise resistance training is performed on the financial data to extract the robust features, and the actor–critic algorithm is based on advantage actor–critic (A2C) and consists of two networks in which the actor network learns and implements the investment policy, which is then evaluated by the critic network to determine the best action plan by continuously redistributing various portfolio assets to maximize the expected return on investment. All the sample data for this study were collected from Yahoo Finance. We tested our method on 24 Dow Jones stocks with sufficient liquidity by comparing different state preprocessing methods. Experimental results show that the proposed method is effective and better than a SOTA algorithm. The contributions of this work are three-fold: (1). we propose a state preprocessing network that employs SSDAE to perform anti-noise training on stock data to extract the robustness features so as to improve the robustness of agent transaction decision making; (2). we have performed extensive experiments to test the performance of our proposed method and demonstrate that it outperforms other state preprocessing methods, such as PCA&DWT [7], LSTM-AE [10], RSAE, Dow Jones Industrial Average index (DJIA), and a SOTA method [11]; and (3). the experimental results show that our proposed method performs better in balancing benefits and risks and demonstrates the importance of state preprocessing in reinforcement learning trading.

2. Preliminary

2.1. Problem Setup

We formulate the portfolio trading process as a generalized Markov decision process (MDP), and the stock market state was reconstructed using the autoencoder technology. MDP is defined as the tuple $(\mathcal{S},\ \mathcal{A},\ p,\ r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p(s_{t+1}|s_t, a_t)$ represents the probability of selecting action $a_t \in \mathcal{A}$ from $s_t \in \mathcal{S}$ to the next state $s_{t+1} \in \mathcal{S}$, and $r(s_t, a_t, s_{t+1})$ represents the direct reward of taking action at state s and arriving at the new state s_{t+1}. The goal of RL agent is to maximize the cumulative income while controlling the transaction cost and risk cost.

The goal was to find an optimal policy π^*:

$$\pi^* = \underset{\pi}{\arg max} \sum_{t=0}^{T} \mathrm{E}_{(s_t,a_t)\sim\rho_\pi}\left[\gamma^t r(s_t, a_t)\right] \tag{1}$$

where ρ_π indicates the distribution of state–action pairs that the RL agent will encounter under the control of policy π.

For each policy π, you can define its corresponding Q-value function:

$$V_\pi(s_t) = \mathrm{E}_{a_t\sim\pi}[Q_\pi(s_t, a_t)] \tag{2}$$

2.1.1. State Space

The state space of our trading environment consisted of four parts $[b_t, h_t, p_t, X_t]$. Each letter is defined as follows:

- $b_t \in \mathbb{R}_+$:available balance at current time step t.
- $h_t \in \mathbb{Z}_+^n$: shares owned of each stock at current time step t.
- $p_t \in \mathbb{R}^n$: close price of each stock at current time step t.
- $X_t \in \mathbb{R}^n$: encoding features at current time step t.

2.1.2. Action Space

The agent's job in the portfolio trading problem is to figure out how much of each stock to buy and sell. We do not enable investors to short assets in our action space; we do not allow investors to borrow assets and then return them in the future; we do not allow investors to borrow assets and then return them in the future. For a single stock, our action space was defined as $\{-k, -1, 0, 1, \ldots, k\}$, where k and $-k$ represent the number of shares we can buy and sell, and $k \leq h_{\max}$.$h_{\max}$ is a predefined parameter that sets the maximum number of shares for each buying action; the action space was normalized to $[-1, 1]$, which also means that our action space was continuous, and after the action selection of each state, we executed the sell action first and then the buy and hold action.

2.1.3. Reward Function

The reward function is the change in portfolio value caused by action in states s_t and s_{t+1}.

$$r(s_t, a_t, s_{t+1}) = R_{t+1} - R_t \tag{3}$$

where $R_t = b_{t+1} + p_{t+1}^T h_{t+1} - c_t$ denotes the portfolio values at $t + 1$ and t, respectively. The portfolio value is the total stock value plus the balance, and c_t is the transaction cost.

The reward function can also be set directly to the trade return [12,13], as follows:

$$Tr_t^s = ln\left(\frac{[R_t - R_{t-s}]}{R_{t-s}}\right) \tag{4}$$

where Tr_t^s represents the realized logarithmic rate of the trade return in a period of time t, and the length of the period is s.

Each deal entails transaction charges, which come in a variety of forms and are charged differently by different brokers. We assume our transaction cost is $c_t = p^T \sum_{d=1}^{D} |k_t| \times \lambda$. We also use the existing financial technical indicators to trade off the return and risk, such as the Sharpe ratio [1,14,15].

$$SHr_t = \frac{Tr_t^s - r_f}{(\hat{\sigma_t})^2} \tag{5}$$

where $Tr_t^s - r_f$ is the excess return in a period of trading; $(\hat{\sigma_t}) = \sqrt{252} \times \frac{1}{n-1} \sum_{t=i}^{n+i} (Tr_t^1 - \frac{1}{s} Tr_t^s)$ is the standard deviation of the daily trade return multiplied by $252^{1/2}$ (252 is the approximate number of trading days in a year and is the frequently used factor in the financial field), $i = \{1, \ldots, T\}$; r_f is the risk-free rate of return; and Tr_t^1 is the daily return.

2.2. Trading mode

Assuming that we have \$1 million and trade in D stocks, each stock in the portfolio will be bought gradually, and the remaining funds after daily trading will be the balance b.

The trading environment assumptions are:

- The trading volume of the agent is very small compared to the size of the whole market, so the agent's trades do not affect the state transition of the market environment.
- The liquidity of the market is high enough that the orders can be rapidly executed at the close price.
- The transaction cost is a fixed proportion λ of the trade volume of the day.

3. Methodology

In this section, we present how we depicted stock markets, an overview of SSDAE, and an optimized method of DRL.

3.1. Depiction of Stock Markets

The stock market has the characteristics of a complex description, being non-stationary, and low signal–to–noise ratio. The key issue in stock trading is to grasp the correct trading

moment and perform the correct trading behavior according to the market situation. As for the stock market, the commonly employed data is regular sequences, such as open price, close price, high price, low price, and volume (OHCLV) data. In order to fully describe the market, we added financial technical indicators. The technical indicators are very close to the market, respond quickly to the short-term changes of the market, and are intuitive and clear. The technical indicators that are currently popular in the stock market can be classified into five categories: moving averages, volatility, trend, momentum, and volume. Our initial features consisted of 11 + 5 features selected from the above five categories, as well as OHCLV data. Table 1 shows all the technical indicators we used.

Table 1. Summary of financial technical indicators.

Type	Financial Technical Indicators
Moving averages	Simple Moving Average (SMA)
Moving averages	The Exponential Moving Average (EMA)
Volatility	Average True Range (ATR)
Volatility	Williams Percent Range (WPR)
Trend	Moving Average Convergence Divergence (MACD)
Trend	Commodity Channel Index (CCI)
Momentum	Relative Strength Index (RSI)
Momentum	Awesome Oscillator (AO)
Momentum	TSI Indicator (TSI)
Volume	Force Index (FI)
Volume	On-Balance Volume (OBV)

Simple moving average (SMA) is a fundamental technical indicator that is defined as

$$SMA_t = (c_{t-n} + \cdots + c_{t-1} + c_t)/n \tag{6}$$

where c_t represents the close data over a given time period and n represents the time span.

The exponential moving average (EMA) is a trend-following indicator that emphasizes recent data above historical data. It is defined as an exponentially decreasing weighted moving average as

$$EMA_n(c_t) = \frac{2c_t + (n-1)EMA_n(c_{t-1})}{n+1} \tag{7}$$

The moving average convergence divergence (MACD) is an indicator that indicates the difference between a long-term moving average and a short-term moving average. This indicator keeps the advantages of moving averages while also avoiding the generation of erroneous trade signals. *MACD* can be expressed as follows:

$$MACD = 2 * (DIF - DEA) \tag{8}$$

$$DIF_t = EMA(12)_t - EMA(26)_t \tag{9}$$

where *DEA* is the arithmetic mean of *DIF* over the time span. and *EMA* is defined in Equation (7).

The Williams percentage range (WPR) is a dynamic technical indicator to judge whether the market is overbought/oversold. WPR is very similar to the random oscillator. The only difference is that the scale of WPR is from top to bottom, while the random oscillator has internal smoothing.

$$WPR = \frac{(HH_{t-n} - c_t)}{(HH_{t-n} - LL_{t-n})} \times 100 \tag{10}$$

where HH_{t-n} and LL_{t-n} represent the maximum value of the highest price and the minimum value of the lowest price in the past n time intervals, respectively.

The relative intensity index (RSI) is a price-following oscillator:

$$RSI = 100 - 100/\left(1 + \frac{\sum_{i=0}^{n-1} u_{t-i}}{n} / \frac{\sum_{i=0}^{n-1} d_{t-i}}{n}\right) \tag{11}$$

where u_t represents the upward price change at time t, and d_t represents the downward price change at time t.

The commodity channel index (CCI) measures the deviation between commodity prices and their average statistical prices. The higher the index, the higher the price than the average. The lower the price, the lower the average price.

$$CCI = (TP_t - SMA\ (TP,\ n))/0.015MD \tag{12}$$

where TP (typical price) represents the typical price, and its calculation formula is $TP_t = (h_t + l_t + c_t)/3$; h_t represents the highest price at time t; l_t represents the lowest price at time t; MD (mean deviation) is the average deviation of TPl and v_t represents the trading volume at time t.

The on-balance volume (OBV) is to quantify the trading volume, make a trend line, cooperate with the stock price trend line, and speculate on the market atmosphere from the change of price and the increase or decrease of trading volume.

$$\mathcal{O}BV_t = \begin{cases} \mathcal{O}\mathcal{B}V_{t-1} + v_t, & if\ c_t > c_{t-1} \\ \mathcal{O}\mathcal{B}V_{t-1} - v_t\ , & if\ c_t < c_{t-1} \\ \mathcal{O}\mathcal{B}V_{t-1}\ \ , & \text{otherwise} \end{cases} \tag{13}$$

where v_t represents the trading volume at time t.

The awesome oscillator (AO) calculates the difference between the simple moving average of phase 34 and phase 5. The simple moving average used is not calculated using the closing price but using the midpoint of each line. The AO is usually used to identify trends or predict possible reversals.

$$AO = SMA(MEP,\ 5) - SMA(MEP,\ 34) \tag{14}$$

where MEP is median price, and the formula is MEP = (high + low)/2

The force index (FI) is used to indicate the strength of an upward or downward trend. The index measures the strength of bull potential at each increase and bear potential at each decrease.

$$FI = EMA((c_t - c_{t-n}) * v_t, n) \tag{15}$$

The true strength index (TSI) is a technical momentum oscillator used to identify trends and reversals. The indicator may be useful for determining overbought and oversold conditions, indicating potential trend direction changes via centerline or signal line crossovers and warning of trend weakness through divergence.

$$TSI(c_t, r, s) = \frac{EMA(EMA(mtm, r), s)}{EMA(EMA(|mtm|, r), s) \times 100} \tag{16}$$

where $mtm = c_t - c_{t-1}$, r is usually set to 25, and s is set to (12).

3.2. Overview of SSDAE

Raw financial data can only indicate patterns or characteristics of market dynamics to a limited extent, so it is not ideal for direct input to RL agents. However, putting all price data and technical indicators into the state space will lead to complex design and delay. Delay is a key factor in algorithmic trading. Most studies simplify the state space by using only OHCLV data [14]. A better solution is to reduce the dimension through the feature extraction method. Therefore, we propose using an autoencoder network to extract features

from raw financial data and encoder features with robustness information. Figure 1 depicts the SSDAE autoencoder network's structure. x_i indicates the input data of the ith node, and $h_k^{(i)}$ indicates the input data of the kth node of the Nth hidden layer. The arrow in the network represents the connection weight between two adjacent layer nodes.

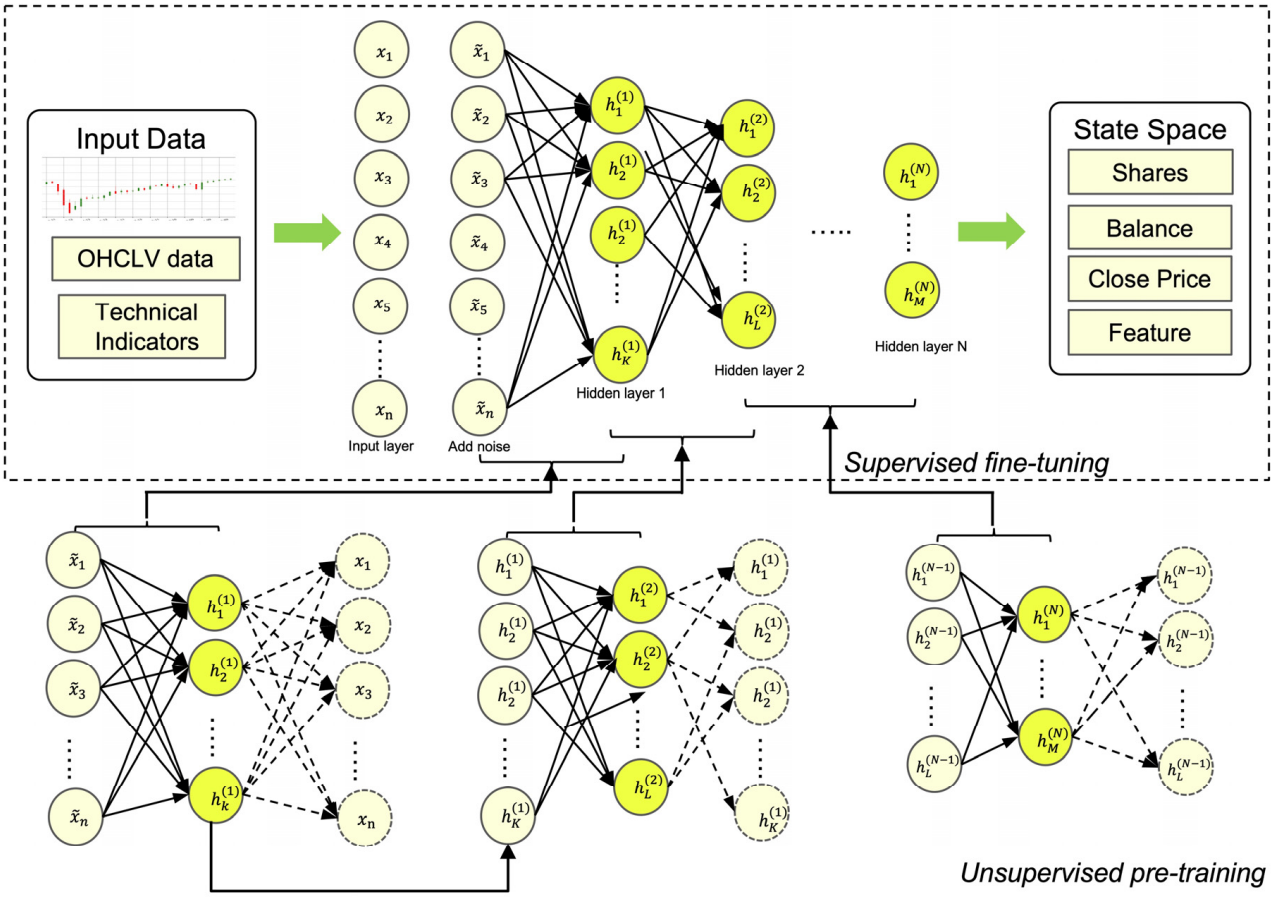

Figure 1. The structure of SSDAE autoencoder network.

3.2.1. Sparse Denoising Autoencoder (SDAE)

Considering that the stock market is also influenced by external factors such as social networks, politics, and natural disasters (such as COVID-19); in order to make RL agents more robust after training, we naturally introduced denoising autoencoder (DAE)) [15] and sparse autoencoder (SAE) to reconstruct original financial data, the first to eliminate accidental events and the second to minimize external factors.

The Autoencoder (AE) [16] is a type of unsupervised machine learning; the AE can be regarded as a three-layer neural network. The feedforward functions of the AE can be defined as follows:

$$\begin{aligned} y &= f_\theta(x) = s(Wx + b) \\ z &= g_\theta(y) = s(W'y + b') \end{aligned} \tag{17}$$

where $s(x)$ is the activation function. In this paper, we use the sigmoid function $s(t) = (1 + \exp(-t)) - 1$; W is the weight matrix; b is the bias vector; and W', b', W, and b are initialized with random values.

The AE only relies on the strategy of minimizing reconstruction errors for network training, which may cause the network to learn only the copy of the original input. Considering that the stock market is also influenced by external factors such as social networks, politics, and natural disasters (such as COVID-19), in order to make RL agents more robust after training, we naturally introduced a denoising autoencoder (DAE) [15] and sparse

autoencoder (SAE) to reconstruct original financial data; the first is to eliminate accidental events, and the second is to minimize external factors.

Based on the AE, DAE adds noise to the input layer data (randomly zeroing the input layer nodes) to make the whole encoder network more robust. It passes on the original input vector x to noise processing result $\widetilde{x}$. The random selection unit was forcibly set to 0 according to a certain proportion, and then used the data $\widetilde{x}$, so as to train better anti-noise ability.

The SAE adds conditional restrictions on the AE, $\hat{\rho}_j = \rho$. The network units were randomly activated to prevent the encoder from working with a low reconstruction error at each point, which also makes the whole encoder network more robust. ρ is the sparsity parameter (generally a smaller value close to 0). In order to achieve this limitation, the objective function introduced an additional penalty factor, namely relative entropy (Kullback–Leibler divergence), which is a method to measure the difference between two distributions. The principle is shown in:

$$\mathrm{KL}(\rho \parallel \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{18}$$

where KL is expressed in ρ, $\hat{\rho}_j$ is the relative entropy between two Bernoulli random variables with mean value. When $\hat{\rho}_j = \rho$, $\mathrm{KL}(\rho \parallel \hat{\rho}_j) = 0$, and the size of KL increases with the difference between ρ and $\hat{\rho}_j$ increasing monotonically; and when $\hat{\rho}_j$ approaches 0 or 1, the relative entropy becomes infinite, so minimizing the penalty factor can make $\hat{\rho}_j$ approach ρ.

Given the raw financial training data set $D = \{(x_i, y_i)\}$, where $i = 1, 2, \ldots, N$ and N denotes the total number of training samples, the SDAE was trained by a backpropagation algorithm to minimize the following sparsity regularized reconstruction loss function:

$$J_{\mathrm{SDAE}}(\boldsymbol{w}, \boldsymbol{b}) = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \parallel h_{w,b}(\widetilde{x})^{(i)} - y^{(i)} \parallel_2^2 \right) + \beta \sum_{j=1}^{s} \mathrm{KL}(\rho \parallel \widetilde{\rho}_j) + \frac{\lambda}{2} \left(\parallel W \parallel_F^2 + \parallel W' \parallel_F^2 \right) \tag{19}$$

where β is the sparsity term parameter, and λ is the weight decay term parameter.

3.2.2. Stacked Sparse Denoising Autoencoders (SSDAE)

To enhance the learning ability of the autoencoder network, the method of increasing the number of network layers was generally adopted. The error back propagation algorithm was used in the majority of our existing shallow neural networks, including ordinal autoencoder networks. When the number of layers is increased, the propagation error decreases, and the gradient disappearance problem becomes more frequent. This issue was avoided by using a stacked autoencoder. Figure 1 shows the SSDAE network structure of N hidden layers. At present, the SSDAE network still adopts the greedy layer-by-layer training method [16], which mitigates the step dispersion phenomena to a certain extent. That is, the first hidden layer is pre-trained. Then, the output of the first layer is used to pre-train the second layer. This procedure is repeated for the remaining layers. After all layers have been pre-trained, the network used for encoding in each layer is erased, leaving only the decoding network of each layer. Then, using the BP algorithm, all layers are fine-tuned until the network's ideal parameters are reached. The SSDAE network converts the original input into the nth layer's extracted features. The SSDAE network's loss function during the fine-tuning stage is as follows:

$$J_{\mathrm{SSDAE}}(\boldsymbol{w}, \boldsymbol{b}) = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \parallel h_{w,b}(\widetilde{x})^{(i)} - y^{(i)} \parallel_2^2 \right) + \frac{\lambda}{2} \sum_{l=1}^{2l} \left(\parallel W_l \parallel_F^2 \right) \tag{20}$$

where W_l is the weight of the lth layer in the SSDAE. The sparsity term was omitted because the pretrained weights were employed as regularization to our network.

Considering that some studies have focused on the accuracy of AE-predicted and -reconstructed financial sequence data [17–21], and the effectiveness of feature extraction can only be judged according to the results of strategy performance, we ignored the accuracy verification of AE reconstruction data in this paper.

3.3. Optimization via Reinforcement Learning—Advantage Actor–Critic Learning

There are currently only a handful of quantitative studies that employ the actor–critic algorithm. The authors of [11] conducted a comparison of three actor–critical algorithms: proximal policy optimization (PPO), advantage actor critical (A2C), and deep deterministic policy gradient (DDPG). Experimental results indicated that A2C outperforms other algorithms. As a result, this paper employed A2C and implemented it using stable-baselines3 [22]. The framework of our proposed model is described in Figure 2.

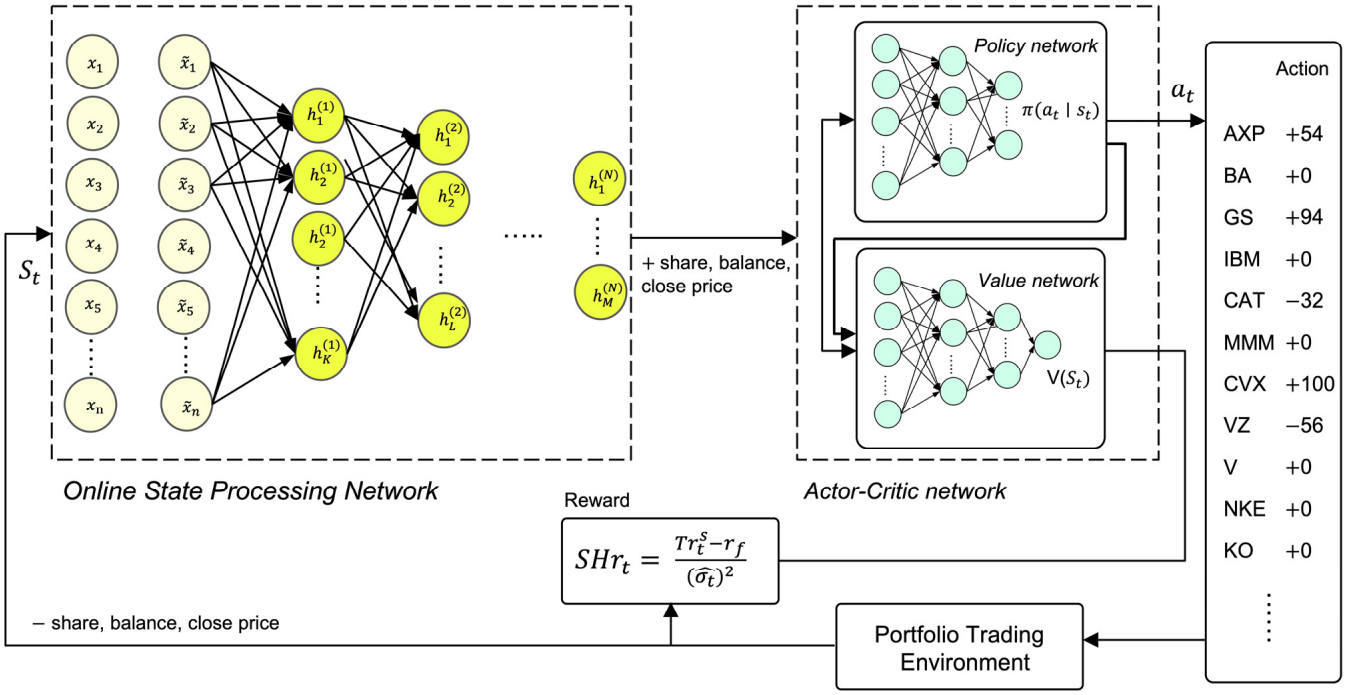

Figure 2. Overview of the proposed model.

The A2C algorithm [23] proposed by OpenAI is the synchronous variant of Google's Asynchronous Advantage Actor Critic (A3C) algorithm [24]. It has been demonstrated that A2C performs similarly to A3C but with less implementation and execution complexity. The A2C is different from other actor–critic algorithms in that it improves the policy gradient updates. A2C uses the advantage function instead of the original return in the critical network, which can be used as an indicator to measure the quality of the selected action value and the average value of all actions. In addition to estimating the advantage function, the critic network also calculates the value function. This means that by evaluating how excellent an action is and how good it can be at the same time, the policy network's large variance is decreased, and the model becomes more robust. In large batch sizes, synchronous gradient updates are more cost-effective, faster, and superior. The A2C's stability properties make it ideal for stock trading.

We updated our policy based on the value function:

$$\nabla J_\theta(\theta) = \mathbb{E}[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A(s_t, a_t) \tag{21}$$

where $\pi_\theta(a_t \mid s_t)$ is the policy network. $A(s_t, a_t)$ is the advantage function and can be written as

$$A(s_t, a_t) = q_\pi(s_t, a_t) - V_\pi(s_t), \quad \text{or} \quad A(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma V_\pi(s_{t+1}) - V_\pi(s_t) \tag{22}$$

where $V_\pi(s)$ is the value expectation of the executor network in a certain state, $q_\pi(s, a)$ is the maximum value obtained by selecting action a in a certain state, $V_\pi(s) \leq q_\pi(s, a)$, and updated policy $\pi'(s) = argmax_a q_\pi(s, a)$. The following is evidence of strategy improvement:

$$\begin{aligned} & v_\pi(s) \leq q_\pi(s,a) = E[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s))] \\ = \; & E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \leq E_\pi, [R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\ = \; & E_\pi[R_{t+1} + \gamma E_\pi, (R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}] | S_t = s] \\ = \; & E_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \middle| S_t = s\right] \\ \leq \; & E_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \middle| S_t = s\right] \cdots\cdots \leq v_{\pi'}(s) \end{aligned} \tag{23}$$

According to the greedy strategy of $a = argmax_a q_\pi(s, a)$, the new strategy obtained by updating all states must make each state's value equal to or greater than its value before the update. As demonstrated above, the strategy of RL must be improved after each round of renewal.

4. Experiments

4.1. Dataset Descriptions

We provide a summary of a real-world datasets of our experiments in Table 2. All the data used in this work is available on Yahoo Finance and Google Finance.

Table 2. Training, validation, and test sets.

Market	Start Date	End Date	Type
The US	2014/01	2019/01	Training set
	2019/01	2020/01	Validation set
	2020/01	2022/01	Test set

4.2. Evaluation Metrics

We used six performance measurement metrics to evaluate the back test results:

- Cumulative return (CR): is the total amount earned by an RL agent during the trading period, excluding transaction costs.
- Maximum drawdown (MDD) [25]: refers to the maximum loss percentage from peak to trough during the trading period.
- Sharp ratio (SR): Sharp ratio [26] is used to measure the rate of return that trading strategies can achieve when confronted with a unit of risk. It is the most commonly used mainstream standardized metric for evaluating portfolio strategies' performance. Equation (5) has illustrated the calculating formula.
- Calmar ratio (CMR): Calmar ratio [27] is used to measure the risk by using the concept of MDD.
- Alpha: Alpha value is used to measure the excess return obtained by the model relative to the benchmark within the trading range. The greater the alpha value, the more capable it is of generating additional returns in comparison to the benchmark. The calculation process of alpha value is shown in:

$$\text{Alpha} = R_p - \left[R_f + \beta_p\left(R_m - R_f\right)\right] \tag{24}$$

where R_p is the yield of the model, β_p is the beta value of the model, and R_m is the yield of the benchmark strategy.

- Beta: Beta value is an indicator used to assess the systemic risk of the model relative to the benchmark. If the beta value is greater than one, the volatility of the model is greater than the benchmark; if the beta value is less than one, the volatility of the model is less than the benchmark; and if the beta value is equal to one, the volatility of the model is the same as that of the benchmark. The computation method is illustrated below:

$$\text{Beta} = \text{Cov}\left(R_p, R_m\right) / \sigma_m^2 \tag{25}$$

where Cov is the covariance, and σ_m^2 is the variance of the benchmark strategy.

CR and alpha values are to evaluate the profitability of RL agent separately; MDD and beta values are used to measure the ability of risk control; SR is a risk adjusted return, which can consider the performance, comprehensive income, and risk of the strategy.

4.3. Experimental Details

Our stock portfolio trading environment was based on OpenAI gym [28]. In order to better test the anti-risk performance of strategies, we chose to test the time set, including the time period of the COVID-19 outbreak. Finally, the parameters that needed to be set in advance are summarized in Tables 3 and 4. To prevent inadequate learning such as local minimum and overfitting, dropout and gradient were applied.

Table 3. Summary of RL agent's parameters.

Parameters	Value
trading window size (s)	20
risk-free rate of return $\left(r_f\right)$	1%
A2C_learning_rate	0.0007
A2C_ n_steps	5

Table 4. Summary of SSDAE parameters.

Structure Parameters	Value	Learning Parameters	Value
Hidden layer 1	12	Batch size	100
Hidden layer 2	9	Epoch of pre-training	50
Hidden layer 3	6	The sparsity	10^{-8}
Hidden layer 4	3	The regularization term	10^{-5}

4.4. Discussion of Comparison Methods

It was also difficult to compare the results of our proposed method with other studies due to the difference in transaction variety, mode, environment, and RL algorithm. All comparative methods are trained and traded under the same trading environment, trading rules, and parameters. For comparison with our proposed model, in addition to the DJIA index, we selected an advanced portfolio management strategy from the current literature to compare the performance that was an open-source method that does not use feature extraction and improves performance by integrating multiple agents [11]. We also implemented several variants or simplified versions of SSDAE to investigate the roles of several key components: we chose RSAE [29], which lacked a sparse and denoise network compared with our proposed method; LSTM-AE [10], a method of extracting time series information for financial series; and a machine learning method using PCA and wavelet denoising technology [7].

4.5. Results and Analysis

As shown in Figure 3 and Table 5, the SSDAE agent performs best in all measurement indicators. It was observed that our proposed method outperforms the DJIA index and the SOTA method without state preprocessing in all measurement indicators. This demonstrates the advanced nature of our proposed method in RL trading algorithm. Compared

with the variants (LSTM-AE, PCA&DWT) or simplified version (RSAE), our method shows that it does better in balancing benefits and risks: In terms of CR and Alpha, the SSDAE agent outperforms other state preprocessing approaches, hitting 88.5% (CR) and 0.28 (alpha). In the beta values and MDD (%), they reached 0.69 and −21.7, respectively. Finally, from the perspective of comprehensive income and risk, there is no doubt that SSDAE agent performs best in terms of SR and CMR, which are 1.37 and 1.71 respectively.

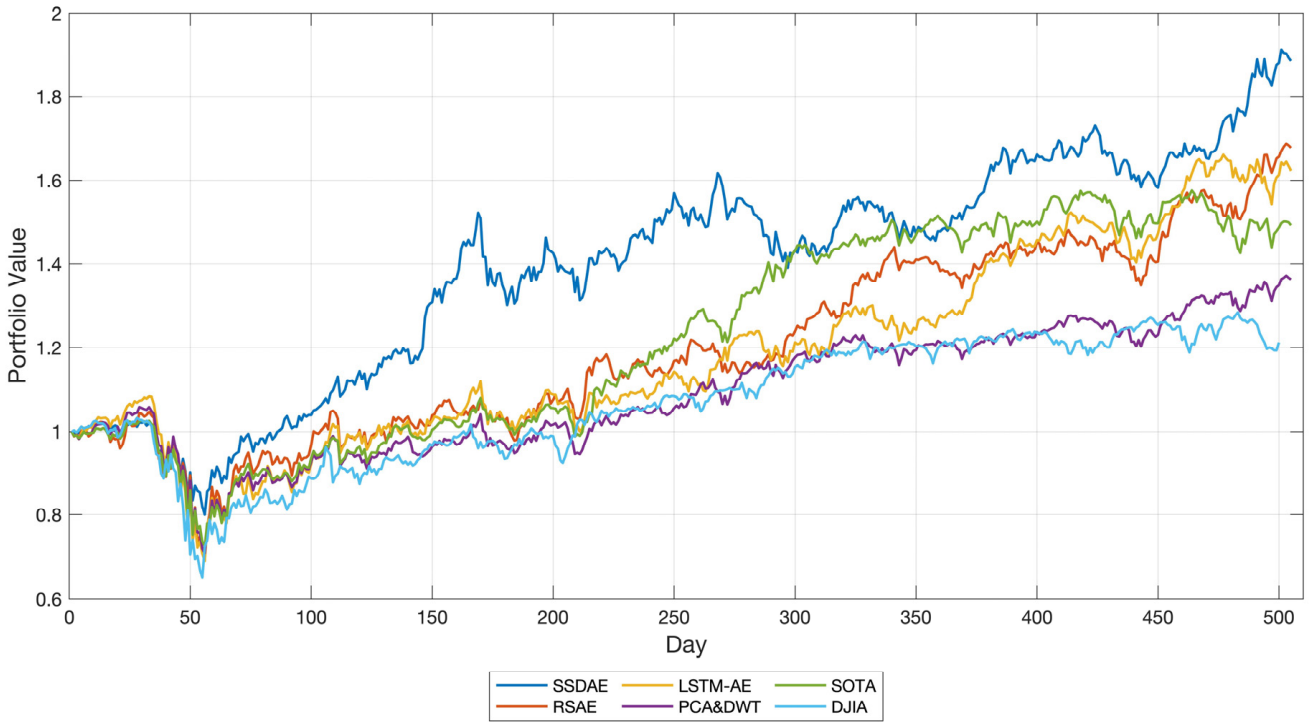

Figure 3. The cumulative wealth on DJIA.

Table 5. The results in different RL methods.

	CR (%)	MDD (%)	SR	CMR	Alpha	Beta
SSDAE	88.5	−21.7	1.37	1.71	0.28	0.69
RSAE [29]	67.7	−33.9	0.99	0.87	0.16	1.01
LSTM-AE [10]	62.3	−36.3	0.91	0.75	0.14	1.06
PCA&DWT [7]	36.3	−33.1	0.71	0.05	0.05	0.94
SOTA [11]	49.1	−29.2	0.89	0.76	0.10	0.90
DJIA	26.9	−37.1	−0.17	0.35	0.00	1.00

The most significant feature of our proposed method was the risk control ability compared with other methods. This shows that extracting robust features from anti-noise training can improve the anti-risk ability of trading strategy. The same as the conclusion of the research on the prediction of financial sequences using an autoencoder method [30], our experiment also found that the use of autoencoder technology can reduce the noise of the features of stock markets. In general, noise resistance training is very important for the RL trading system.

5. Conclusions

In this paper, according to the characteristics of the exogenous nature, complexity, and noise of financial time series data, we proposed an anti-risk portfolio trading system based on DRL. It consists of an SSDAE autoencoder and an advantage actor–critic based RL agent. By comparing the trading performances of different state preprocessing methods, including two variants methods, a simplified version, a SOTA method, and the benchmark

DJIA, we found that the impact of anti-noise training on the trading strategy trained by RL agent is positive and promising. Results also illustrate the importance of state preprocessing in reinforcement learning. The advantages of the proposed method are its expansibility, applicability, flexibility, and speed to train, but its interpretability needs further discussion. We will explore using better risk measurement tools such as value-at-risk [31] and conditional-value-at-risk [32] to design newly effective reward functions and, considering the excessive trading volume of the strategy, optimize the transaction cost in the trading process, which is also an issue that needs attention.

Author Contributions: Conceptualization, Formal Analysis, Data Analysis, Data Interpretation, Literature Search, Software, Methodology, and Writing—Original draft, H.Y.; Developed the contextualization of the state of the art, Conceptualization, Funding Acquisition, and Project Administration, J.L.; Resources, Supervision, Validation, and Writing—review and editing, D.T.; Literature Search, Investigation, Writing—review and editing, Proofreading, and Visualization, Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by [the National Social Science Foundation of China] grant number [18BGL224] and the APC was funded by [the National Social Science Foundation of China] grant number [18BGL224].

Data Availability Statement: The stock price and trading volume data that support the findings of this study are available from Yahoo Finance [32].

Acknowledgments: We acknowledge the financial support from the National Social Science Foundation of China (18BGL224).

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

References

1. Almahdi, S.; Yang, S.Y. An Adaptive Portfolio Trading System: A Risk-Return Portfolio Optimization Using Recurrent Reinforcement Learning with Expected Maximum Drawdown. *Expert Syst. Appl.* **2017**, *87*, 267–279. [CrossRef]
2. Bertoluzzo, F.; Corazza, M. Testing Different Reinforcement Learning Configurations for Financial Trading: Introduction and Applications. *Procedia Econ. Financ.* **2012**, *3*, 68–77. [CrossRef]
3. Deng, Y.; Bao, F.; Kong, Y.; Ren, Z.; Dai, Q. Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 653–664. [CrossRef] [PubMed]
4. Fischer, T.G. *Reinforcement Learning in Financial Markets—A Survey*; FAU Discussion Papers in Economics: Erlangen, Germany, 2018.
5. Jiang, Z.; Liang, J. Cryptocurrency Portfolio Management with Deep Reinforcement Learning. In Proceedings of the 2017 Intelligent Systems Conference (IntelliSys), London, UK, 7–8 September 2017; IEEE: New York, NY, USA, 2017; pp. 905–913.
6. Zhang, Y.; Zhao, P.; Li, B.; Wu, Q.; Huang, J.; Tan, M. Cost-Sensitive Portfolio Selection via Deep Reinforcement Learning. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 236–248. [CrossRef]
7. Li, L. An Automated Portfolio Trading System with Feature Preprocessing and Recurrent Reinforcement Learning. *arXiv* **2021**, arXiv:2110.05299.
8. Qi, Y.; Wang, Y.; Zheng, X.; Wu, Z. Robust Feature Learning by Stacked Autoencoder with Maximum Correntropy Criterion. In Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014; IEEE: New York, NY, USA, 2014; pp. 6716–6720.
9. Li, W.; Shang, Z.; Gao, M.; Qian, S.; Zhang, B.; Zhang, J. A Novel Deep Autoencoder and Hyperparametric Adaptive Learning for Imbalance Intelligent Fault Diagnosis of Rotating Machinery. *Eng. Appl. Artif. Intell.* **2021**, *102*, 104279. [CrossRef]
10. Jung, G.; Choi, S.-Y. Forecasting Foreign Exchange Volatility Using Deep Learning Autoencoder-LSTM Techniques. *Complexity* **2021**, *2021*, 6647534. [CrossRef]
11. Yang, H.; Liu, X.-Y.; Zhong, S.; Walid, A. Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. In Proceedings of the First ACM International Conference on AI in Finance, New York, NY, USA, 15–16 October 2020; pp. 1–8.
12. García-Galicia, M.; Carsteanu, A.A.; Clempner, J.B. Continuous-Time Reinforcement Learning Approach for Portfolio Management with Time Penalization. *Expert Syst. Appl.* **2019**, *129*, 27–36. [CrossRef]
13. Xu, K.; Zhang, Y.; Ye, D.; Zhao, P.; Tan, M. Relation-Aware Transformer for Portfolio Policy Learning. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, Yokohama, Japan, 7–15 January 2021; pp. 4647–4653.
14. Théate, T.; Ernst, D. An Application of Deep Reinforcement Learning to Algorithmic Trading. *Expert Syst. Appl.* **2021**, *173*, 114632. [CrossRef]
15. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.-A. Extracting and Composing Robust Features with Denoising Autoencoders. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 1096–1103.

16. Hinton, G.E.; Osindero, S.; Teh, Y.-W. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef] [PubMed]
17. Mohanty, D.K.; Parida, A.K.; Khuntia, S.S. Financial Market Prediction under Deep Learning Framework Using Auto Encoder and Kernel Extreme Learning Machine. *Appl. Soft Comput.* **2021**, *99*, 106898. [CrossRef]
18. Bi, Q.; Yan, H.; Chen, C.; Su, Q. An Integrated Machine Learning Framework for Stock Price Prediction. In Proceedings of the China Conference on Information Retrieval, Xi'an, China, 14–16 August 2020; Springer: Cham, Switzerland, 2020; pp. 99–110.
19. Bao, W.; Yue, J.; Rao, Y. A Deep Learning Framework for Financial Time Series Using Stacked Autoencoders and Long-Short Term Memory. *PLoS ONE* **2017**, *12*, e0180944. [CrossRef] [PubMed]
20. Xu, Y.; Chhim, L.; Zheng, B.; Nojima, Y. Stacked Deep Learning Structure with Bidirectional Long-Short Term Memory for Stock Market Prediction. In Proceedings of the International Conference on Neural Computing for Advanced Applications, Shenzhen, China, 3–5 July 2020; Springer: Singapore, 2020; pp. 447–460.
21. Gündüz, H. Stock Market Prediction with Stacked Autoencoder Based Feature Reduction. In Proceedings of the 2020 28th Signal Processing and Communications Applications Conference (SIU), Gaziantep, Turkey, 5–7 October 2020; IEEE: New York, NY, USA, 2020; pp. 1–4.
22. Ross, S.; Mineiro, P.; Langford, J. Normalized Online Learning. *arXiv* **2013**, arXiv:1305.6646.
23. Zhang, Y.; Clavera, I.; Tsai, B.; Abbeel, P. Asynchronous Methods for Model-Based Reinforcement Learning. *arXiv* **2019**, arXiv:1910.12453.
24. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
25. Magdon-Ismail, M.; Atiya, A.F. Maximum Drawdown. *Risk Mag.* **2004**, *17*, 99–102.
26. Sharpe, W.F. Mutual Fund Performance. *J. Bus.* **1966**, *39*, 119–138. [CrossRef]
27. Young, T.W. Calmar Ratio: A Smoother Tool. *Futures* **1991**, *20*, 40.
28. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai Gym. *arXiv* **2016**, arXiv:1606.01540.
29. Soleymani, F.; Paquet, E. Financial Portfolio Optimization with Online Deep Reinforcement Learning and Restricted Stacked Autoencoder—DeepBreath. *Expert Syst. Appl.* **2020**, *156*, 113456. [CrossRef]
30. Sun, H.; Rong, W.; Zhang, J.; Liang, Q.; Xiong, Z. Stacked Denoising Autoencoder Based Stock Market Trend Prediction via K-Nearest Neighbour Data Selection. In Proceedings of the International Conference on Neural Information Processing, Guangzhou, China, 14–18 November 2017; Springer: Cham, Switzerland, 2017; pp. 882–892.
31. Jorion, P. *Value at Risk: The New Benchmark for Managing Financial Risk*; McGraw-Hill: New York, NY, USA, 2000.
32. Rockafellar, R.T.; Uryasev, S. Conditional Value-at-Risk for General Loss Distributions. *J. Bank. Financ.* **2002**, *6*, 1443–1471. [CrossRef]