Expert Systems With Applications 218 (2023) 119556

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Deep reinforcement learning for stock portfolio optimization by connecting with modern portfolio theory

Junkyu Jang [1,2], NohYoon Seong [*]

Management Engineering Department, College of Business, Korea Advanced Institute of Science and Technology, Seoul, Korea

ARTICLE INFO

Keywords:
Portfolio management
Modern portfolio theory
Deep reinforcement learning
Technical analysis
Tensor decomposition

ABSTRACT

With artificial intelligence and data quality development, portfolio optimization has improved rapidly. Traditionally, researchers in the financial market have utilized the modern portfolio theory for portfolio optimization; however, with the recent development of artificial intelligence, attempts to optimize portfolios with reinforcement learning are increasing. Many studies have developed reinforcement learning and deep learning algorithms and conducted portfolio optimization research. However, in reality, thus far, the securities industry thus has used the modern portfolio theory, which is sufficiently valuable. Nevertheless, to the best of our knowledge, there has yet to be an attempt to combine modern portfolio theory and reinforcement learning. To bridge this gap in the literature, we propose a novel deep reinforcement learning approach that combines the modern portfolio theory and a deep learning approach. As far as we know, we are the first to combine recent deep learning technology and traditional financial theory. Specifically, we solved the multimodal problem through the Tucker decomposition of a model with the input of technical analysis and stock return covariates. The results show that the proposed method outperforms state-of-the-art algorithms regarding the Sharpe ratio, annualized return, and maximum drawdown. In addition, the proposed method dynamically changes the weight according to the market trend, unlike other state-of-the-art algorithms.

1. Introduction

Analyses and accurate forecasts of stock markets have become increasingly challenging and advantageous (Nam & Seong, 2019). The number of attempts to forecast and optimize portfolios in the financial market has increased owing to the need for a deeper understanding of data analysis and artificial intelligence and the increased amount of data in the financial market. Predicting stock market trends helps traders make appropriate decisions and improve profitability, thereby decreasing possible losses (Seong & Nam, 2021; Song, Liu, & Yang, 2017). Furthermore, portfolio optimization with artificial intelligence prevents behavioral economic biases (Hirshleifer, 2015). Therefore, portfolio optimization is essential in financial decision-making, investment management, and algorithmic trading.

Traditional portfolio optimization studies are based on the modern portfolio theory (Markowitz, 1952). Based on the covariance matrix of historical returns, the modern portfolio theory allocates the asset with the best return relative to the risk. However, recent portfolio optimization studies are based on technical analyses and reinforcement learning (Aboussalah & Lee, 2020; Du et al., 2020).

Technical analysis is a method of predicting stock prices using historical data, including open price, high price, low price, close price, and volume size. For technical analysis, most studies compute technical indicators (TIs), which are mathematical calculations based on historical prices and volume to predict the financial market (Atsalakis & Valavanis, 2009; Zhang et al., 2014). Reinforcement learning defines the environment (i.e., stock market), agent (i.e., trader), and reward (i.e., return, risk, and Sharpe ratio) and optimizes the policy network for maximizing the reward (Aboussalah & Lee, 2020; Du et al., 2020). Jiang, et al. (2017) optimized the portfolio using a convolutional neural network-based policy network with a technical analysis and deterministic policy gradient. Aboussalah and Lee (2020) optimized a portfolio using technical analysis and recurrent reinforcement learning.

Surprisingly, to the best of our knowledge, no attempt has been made to combine a reinforcement learning-based approach with modern portfolio theory. The deep learning reinforcement-based approach uses

* Corresponding author at: Management Information Systems, Business School, KAIST, 85, Heogi-ro, Cheongnyangri-dong, Dongdaemun-gu, Seoul, Korea.
E-mail addresses: jbkjsm@kaist.ac.kr (J. Jang), nyseong@kaist.ac.kr (N. Seong).
[1] Present address: Management Information Systems, Business School, KAIST, 85, Heogi-ro, Cheongnyangri-dong, Dongdaemun-gu, Seoul, Korea.
[2] ORCID: https://orcid.org/0000-0002-7423-0090.

https://doi.org/10.1016/j.eswa.2023.119556
Received 3 February 2022; Received in revised form 20 November 2022; Accepted 12 January 2023
Available online 13 January 2023

technical analysis as a feature, whereas the modern portfolio theory uses correlation in different dimensions, making it challenging to handle together. To bridge this gap in the literature, we combined the reinforcement learning-based approach with modern portfolio theory.

To connect reinforcement learning with modern portfolio theory, we used the tensor decomposition approach, a method of processing multimodal data, to combine the inputs of two different dimensions. From this tensor, the policy network performs feature extraction with a 3D convolutional neural network, and Tucker decomposition is performed on the result tensor to form the core tensor. We then used a deep network for the core tensor to obtain the final output.

To justify the effectiveness of the proposed method, we obtained 29 historical stock datasets from January 2008 to December 2019. The results are as follows. First, the proposed method shows superior performance to state-of-the-art algorithms and base strategies regarding the Sharpe ratio, maximum drawdown, and final accumulative portfolio value, whereas state-of-the-art algorithms show similar performance to base strategies. Second, the proposed method appropriately allocates assets according to the market trend, whereas state-of-the-art algorithms do not significantly change the asset allocation according to the trend. It suggests that the proposed method adapts faster to market changes than state-of-the-art algorithms.

Our contributions as follows:

1. Method: We proposed a portfolio optimization method based on reinforcement learning that incorporates modern portfolio theory. We use DDPG to quickly train a continuous action space to learn various technological features based on modern portfolio theory. We have shown that properly learning to use these features can effectively rebalance portfolios.
2. Theory: We use Tucker decomposition to encode temporal and spatial relationships (multimodal relationships) between the technological features of each asset and dynamic asset allocation using core features (core matrix) obtained from decomposition. We showed that using such a tucker decomposition reduces the number of parameters due to different tech indicators (TIs) over time for each asset and helps in asset rebalancing.
3. Experiments: We backtested the assets included in the Dow Jones Indices, which are applied to many ETFs. Our model using Tucker decomposition showed state-of-the-art performance compared to various models. We also verified our model by removing our proposed tucker decomposition or TI modules.

The remainder of this paper is organized as follows. Section 2 provides a literature review of related works, reinforcement learning, and tensor decomposition. Section 3 describes the data, technical analysis, deep learning models, reinforcement learning, and evaluation. Section 4 presents the experimental results. Section 5 explains the limitations of this study and future work.

2. Literature review

2.1. Related works

Portfolio management methods based on Modern Portfolio Theory (MPT) have received considerable attention in academia and industry for a long time (Elton & Gruber, 1997; Francis & Kim, 2013). However, as the deep learning method has recently emerged, portfolio management using reinforcement learning is in the spotlight.

Portfolio optimization using reinforcement learning has two major research streams: 1) reward adjustment and 2) model development.

Because the reward function is an objective function that maximizes reinforcement learning, it is adjusted to reduce portfolio risk or perform other functions. Almahdi and Yang (2017) solved a portfolio optimization problem that adjusts the weights of five funds using reinforcement learning with a risk-sensitive reward function. (Aboussalah & Lee, 2020) used a reinforcement learning algorithm that considered continuous action and used the regularized Sharpe ratio as a reward function. Vo, He, Liu, and Xu (2019) suggested a method for socially responsible portfolio optimization. They learned reinforcement learning by weighting the environmental, social, and governance (ESG) rating to the Sharpe ratio and using it as a reward function.

Other studies developed deep learning models that appropriately reflect stock price history while maximizing returns. Ye, Pei, Wang, Chen, Zhu, Xiao, and Li (2020) suggested a method that embeds stock price and financial news together through an encoder structure with a deterministic policy gradient algorithm to maximize returns.(Jiang and Liang (2017) presented a portfolio optimization method using CNN-based networks. Soleymani and Paquet (2020) performed better than simple DNN-based algorithms using a restricted stacked autoencoder and a 2-D CNN network. (Wu, Syu, Lin, & Ho, 2021) suggested an algorithm with two neural networks (CNN and RNN). In this study, we followed the model development research stream.

In addition to these reinforcement learning studies, there are also studies using genetic algorithms. Chen, Lu, Hong, Lin, and Gaeta (2019) suggested a diverse group stock portfolio optimization using a group genetic algorithm with the chromosome representation. Chen et al. (2019) improved the genetic algorithm by utilizing fuzzy logic to tune the parameters in the evolution process dynamically. Chen, Chen, Diaz, and Lin (2021) proposed a group trading strategy portfolio based on a group genetic algorithm to find appropriate stop-loss and take-profit points.

Also, some studies analyzed ROI by performing stock price prediction without using reinforcement learning. Minh, Sadeghi-Niaraki, Huy, Min, and Moon (2018) suggested a novel algorithm to predict stock prices with financial news and sentiment dictionary. They used a two-stream gated recurrent unit network and Stock2Vec algorithm and ensemble them. Ding, Wu, Sun, Guo, and Guo (2020) proposed a multi-scale Gaussian transformer to predict stock movements. Seong and Nam (2022) suggested an algorithm that found a financial network with complex system theory and showed better performance than other state-of-the-art algorithms. They showed that the buy-sell approach with the model output probability is effective.

However, despite the importance of the covariance matrix in portfolio management, the covariance matrix has not been used in portfolio optimization using reinforcement learning. This might be because the input of reinforcement learning is price features (Almahdi & Yang, 2017), which are challenging to use simultaneously. After all, the covariance matrix and dimensionality do not match. To fill this void in the literature, we propose a novel approach that builds a bridge between modern portfolio theory and reinforcement learning.

2.2. Reinforcement learning

Reinforcement learning has been used to learn human learning (Sutton & Barto, 2018). When a person first learns to ride a bicycle, they learn how to ride while riding a bicycle without considering the angular momentum, momentum, and force. Similarly, reinforcement learning does not solve an optimization problem according to a specific rule, but an agent defined in a certain environment recognizes the current state and maximizes the reward among actions.

Q-learning is the basis of reinforcement learning (Hester et al., 2018). Q-learning is based on Q-function. Q-function refers to the reward function agent gets when an agent performs an action a in a specific state s. To update the Q function in Q-learning, use the greedy action $a(t+1)$ that maximizes $Q(S_{t+1})$ to obtain the expectation of the Q function as follows:

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

A reinforcement learning algorithm that adopts a neural network and approximates the above Q function is called a deep Q-network (DQN). With DQN, the state is encoded as a value function. However, the DQN

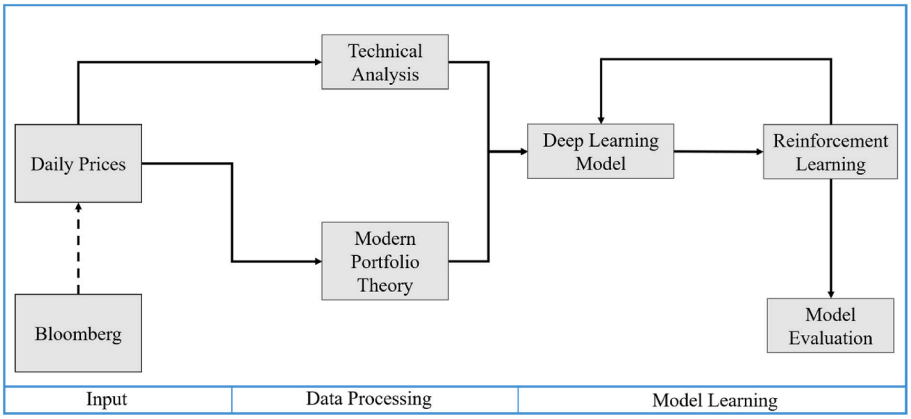

Fig. 1. Proposed approach.

approach is difficult to use because the amount of computation increases when the action space is large (Hester et al., 2018). The portfolio weight in the stock market is not discrete and has an action space in the form of a continuous box, which leads to a curse that infinitely increases the number of action spaces. Therefore, the DQN is not suitable for portfolio optimization.

Many reinforcement-learning algorithms have been used in portfolio management. Some studies have used deep Q-learning (Lucarelli & Borrotti, 2020; Park, Sim, & Choi, 2020). Some studies have used proximal policy optimization (Du et al., 2020; Lin & Beling, 2020). Among them, the deep deterministic policy gradient (DDPG) is widely used (Chaouki, Hardiman, Schmidt, Sérié, & De Lataillade, 2020; Khemlichi, Chougrad, Khamlichi, El Boushaki, & Ali, 2021). DDPG is a reinforcement learning algorithm combining Q-learning and the policy gradient (Li, Wu, Cui, Dong, Fang, & Russell, 2019).

DDPG is an improved version of the deterministic policy gradient algorithm (Chaouki et al., 2020; Khemlichi et al., 2021). First, DPG is an algorithm that finds the optimal policy by combining Q-learning with the policy gradient. In finding such a policy, DDPG uses a neural network to update the parameter using an approximation continuously.

DDPG adopts and uses the off-policy method to update the actor and critical networks separately. The actor-network $\mu(s|\theta\ \mu)$ maps states to actions where $\theta\ \mu$ is the set of actor-network parameters, and the critic network $Q(s, a|\theta)$ outputs the value of action under that state, where θ and Q are the set of critic network parameters. Because exploration for better action in reinforcement learning is essential, random noise N is added and used in the actor-network. In the critic network, the replay buffer R is used to save training and effectively update the model.

In addition, DDPG uses soft-target updates. Soft target update is slowly updating network parameters in updating the actor and critic networks. In a general deep learning method, the parameters of each layer are trained once and continuously updated without saving, but DDPG slowly changes the parameters of the network layer by reusing the learned parameters once. This learning has the advantage of increasing learning stability.

2.3. Tensor decomposition

Tensors are often referred to as high-order generalizations of matrices because they can model more than the two dimensions captured in matrices. In this context, they are also regarded as multi-dimensional arrays: $X \in R^{N_1*N_2*\cdots*N_m}$ where $N_1*N_2*\cdots*N_m$ is the size of the tensor, and M is the number of dimensions, also referred to as the order or number of modes (Sun, Tao, & Faloutsos, 2006).

Rearranging the tensor as a matrix may facilitate the manipulation and processing. As computing tensor is a difficult task because of its complexity, tensor decomposition techniques have been long studied, including Tucker decomposition (Tucker, 1966), canonical decomposition (Carroll & Chang, 1970), parallel factors (Harshman, 1970), and CP decomposition (Kiers, 2000). Tucker decomposition is the most popular approach (Fernandes, Fanaee-T, & Gama, 2021).

Table 1
List of Backtest stocks.

Sector	Name (Ticker)
Technology	Apple Inc. (AAPL)Cisco Systems. Inc. (CSCO)Intel Corporation (INTC)International Business Machines Corporation (IBM) Microsoft Corporation (MSFT) Salesforce.com, Inc (CRM)
Financial Service	Visa Inc. (V)JPMorgan Chase & Co. (JPM)Goldman Sachs (GS)American Express co. (AXP)The Travelers Companies, Inc (TRV)
Consumer Cyclical	Amazon.com, Inc (AMZN)The Walt Disney Company (DIS) The Home Depot, Inc. (HD)The Procter & Gamble co. (PG)
Consumer Defensive	Walmart Inc. (WMT)The Coca-Cola Company (KO) McDonald's (MCD) Nike, Inc. (NKE)
Healthcare	Johnson & Johnson (JNJ)Merck & Co., Inc (MRK)United Health Group Inc. (UNH)Walgreens Boots Alliance, Inc. (WBA)
Industrials	3 M Company (MMM)Honeywell International. Inc (HON) Boeing Co (BA)Caterpillar Inc. (CAT)Chevron Corporation (CVX)
Communication Services	Verizon Communications Inc. (VZ)

Tucker decomposition consists of decomposing a tensor in the (mode) product of a core tensor and a projection/factor matrix for each mode. Mathematically, Tucker decomposition finds a core tensor $y \in R^{R_1*R_2*R_3}$ from a tensor $X \in R^{I*J*K}$ and core tensor size $\{R_1, R_2, R_3\}$ by minimizing $\|X - y*A*B*C\|$, where $A \in R^{I*R_1}$, $B \in R^{J*R_2}$, $C \in R^{K*R_3}$ are factor matrices.

Practically, Tucker decomposition is used to process multimodal data. Ben-Younes, Cadene, Cord, and Thome (2017) used Tucker decomposition to simultaneously process the information on visual cues and text cues in a visual question-answering context. Fonał and Zdunek (2021) presented a method using Tucker decomposition for feature extraction from multimodal data. Peng, Hong, and Zhao (2021) utilized Tucker decomposition when performing sentiment analysis using multimodal data such as audio, visual, and text. In this research stream, we used Tucker decomposition as a method of using time series-based deep learning and modern portfolio theory.

3. Proposed method

This section describes how to construct a system that optimizes a stock portfolio by using the intersection of Modern Portfolio Theory and

Table 2
List of stock returns in the period.

Ticker	MMM	GS	NKE	AXP	HD	PG	AMGN	HON	CRM	AAPL
Train Return(%)	179.0	49.2	240.3	93.6	499.9	63.2	283.4	146.5	305.2	446.4
TestReturn(%)	7.24	−0.4	100.1	72.5	73.8	62.2	73.8	68.3	130.2	162.7
Ticker	**INTC**	**TRV**	**BA**	**IBM**	**UNH**	**CAT**	**JNJ**	**VZ**	**CVX**	**JPM**
Train Return	114.0	217.4	138.7	63.7	412.7	46.3	125.7	140.9	70.9	133.7
Test Return	76.0	20.9	122.9	−9.7	90.4	69.9	35.9	28.3	14.2	71.4
Ticker	**V**	**CSCO**	**MCD**	**WBA**	**KO**	**MRK**	**WMT**	**MSFT**	**DIS**	
Train Return	373.9	38.8	174.4	176.2	79.3	116.1	50.2	146.9	271.8	
Test Return	141.0	71.0	77.3	−23.6	45.8	65.1	86.3	165.7	41.5	

reinforcement learning. First, the data is described. After that, technical analysis, deep learning model for bridging modern portfolio theory and DDPG, reinforcement learning algorithm, are described. Finally, indicators used for evaluation are explained. This series of processes is illustrated in Fig. 1.

3.1. Data

To verify the performance of the proposed algorithm, we backtested the portfolio. The proposed portfolio optimization algorithm is the weight of the stocks in the Dow Jones index. We used 29 stocks, excluding one recently incorporated stock, for data consistency. The indices are listed in Table 1.

The dataset was downloaded from Bloomberg software. The dataset contains daily open, close, high, and low prices, adjusted for stock splits and paid dividends, and the trading volume for the corresponding trading day. The dataset is from January 1, 2008, to December 31, 2019, for every business day. The dataset was divided into two sets: a training set and a test set. The training set contains data from January 1, 2008, to December 31, 2016, and the test set contains data from January 1, 2017, to December 31, 2019.

If there is no data for each stock, the data for each date was filled in using the backfill method. In addition to the given data, we used the Close price data of each stock to calculate and learn technical indicators that traditionally measure the trend of each stock. The technical indicators used in our model are described in Section 3.2.

To intuitively compare our backtest results, we show the stock returns of the training and test periods in Table 2.

3.2. Technical analysis

Technical indicators describe the current state of the market price and incorporate information about its past trends. An indicator can be seen as a snapshot of the current situation that accounts for past behavior over a certain period. To make technical indicators, we need to choose which indicators to use and how far back in the past the indicators impact better predictions of future price movements. We use three leading technical indicators based on the literature: moving average (MA), relative strength index (RSI), and moving average convergence divergence (MACD) (Kara, Boyacioglu, & Baykan, 2011).

MA is a trend-following technical indicator that computes the average price over a window, where P_t is the close price on day t, and w is the window:

$$\mathrm{MA_w(t)} = \frac{\sum_0^{w-1} P_t}{w} \tag{2}$$

RSI is an oscillator that shows the strength or pressure of an asset price movement by comparing the upward or downward movement of an asset's close price, where EMA is an exponential moving average that computes MA with exponential decay weight:

$$RSI_t(w) = 100^*(1 - \frac{1}{1 + \frac{EMA_w(\max(P_t - P_{t-1}, 0))}{EMA_w(\min(P_t - P_{t-1}, 0))}}) \tag{3}$$

$$\mathrm{EMA_w}(t) = \alpha^* P(t) + (1-\alpha)^* EMA_w(t-1), \alpha = \frac{2}{1+w} \tag{4}$$

MACD is a trend-following technical indicator that shows the relationship between the two MAs:

$$MACD_w(t) = \sum_{i=1}^{w} EMA_k(i) - \sum_{i=1}^{w} EMA_d(i) \tag{5}$$

In this paper, for MA, we used a window as 28. For RSI, we used the window as 14, and for MACD, we used k as 26, d as 12, and the window as 9.

3.3. Deep learning model (Policy Network)

We propose a policy network that provides optimal action by connecting Modern Portfolio Theory and reinforcement learning. First, we created an input tensor to connect correlation and historical prices. We used a four-dimensional vector V_t and a four-dimensional correlation matrix Cor_t made up of m-day technical indicators made of n assets as inputs.

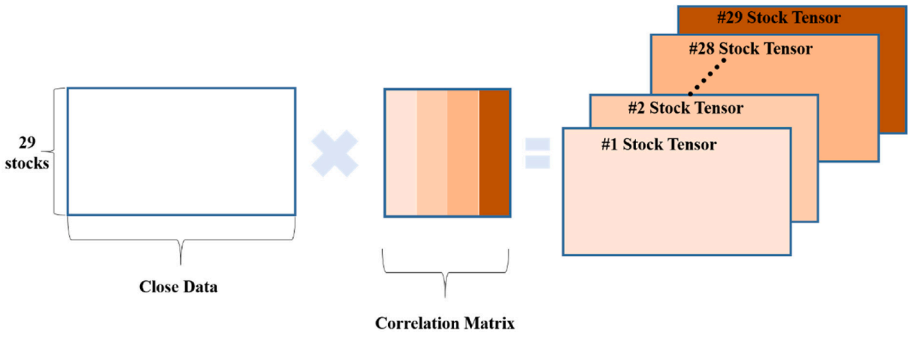

Fig. 2. Input tensors that combine price data and correlation matrix.

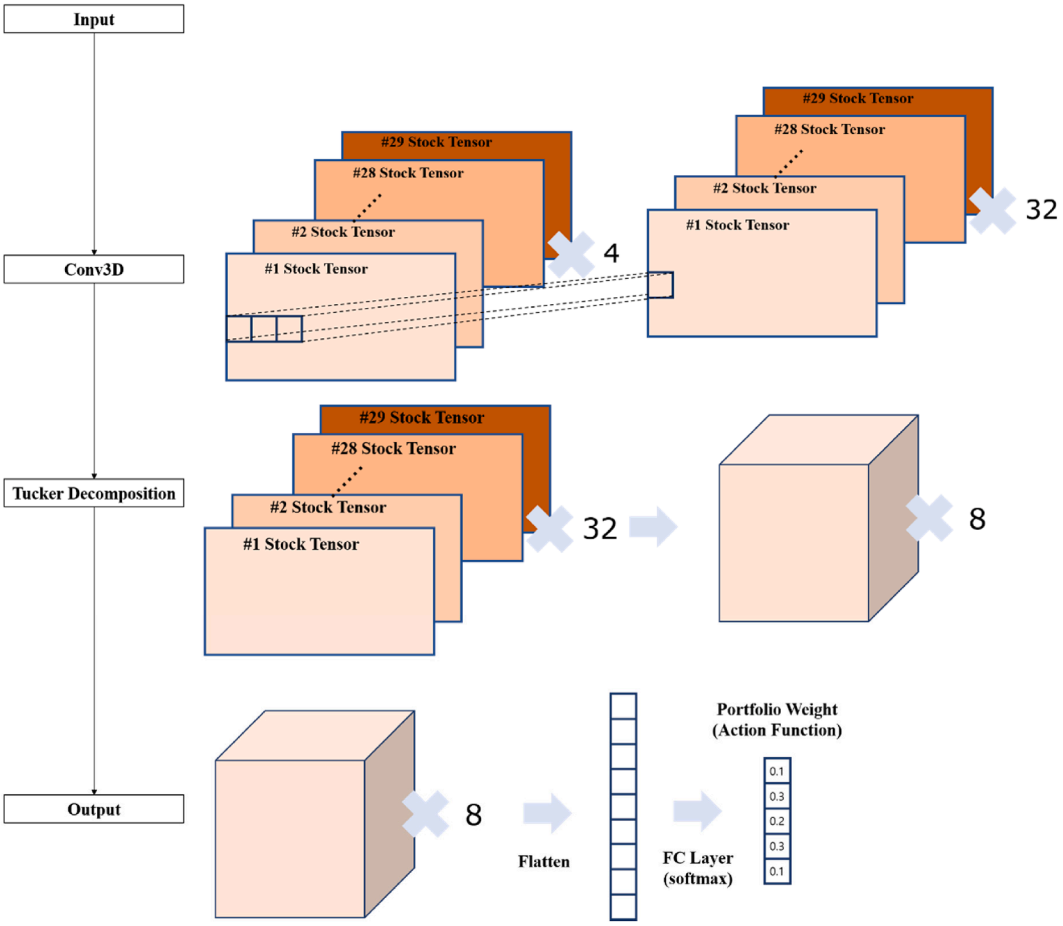

Fig. 3. Policy network architecture.

$$V_t = [V_{t,close}, V_{t,MA}, V_{t,RSI}, V_{t,MACD}] \tag{6}$$

$$Cor_t = [Cor_{t,close}, Cor_{t,MA}, Cor_{t,RSI}, Cor_{t,MACD}] \tag{7}$$

Here, V_t is a tensor composed of technical indicators with a relatively short period of m days, and Cor_t is the relationship between n assets in 252 business days (one year). We use the following transform to fuse the relationship between these two tensors: To combine tensors with two different properties, we perform the following transform for indicator i and asset n, where $\odot$ is the matrix multiplication:

$$F_{t(batch,\ i,n)} = V_{t(batch,i,n)} \odot Cor^{T}_{t(batch,i,n)},\ F_{t(batch,\ i,n)} \in R^{m*n} \tag{8}$$

The transform $F_{t(batch,\ i,j)}$ for any asset j and indicator i means a layer that targets only the part related to the j-th asset for the m*n tensor composed of the indicator. The fusion tensor of the indicator matrix and correlation matrix helps to efficiently model complex relationships between assets by integrating information that can be expressed in short-term and long-term indicators. This process is illustrated in Fig. 2.

The transformed tensor $F_t \in R^{4*n*m*n}$ goes through the process of making four technical indicators into 32 feature maps through the 3D convolution neural network layer with (1, 3, 1) kernel and Relu activation (Agarap, 2018).

$$F'_t = \text{Conv3D}(F_t), F'_t \in R^{32*n*(m-4)*n} \tag{9}$$

We implement the Tucker decomposition on F'_t to decompose a high-dimensional tensor into a lower dimension, which is as follows in equation (11), where C is the core tensor $U_k \in R^{n_k \times R_k}$ is the mode matrix.

$$F'_t[i][i_1, i_2, i_3, i_4] = \sum_{r1=1}^{R_1} \cdots \sum_{r4=1}^{R_4} C[i](r_1, .r_4) U_1(i_1, r1) \cdots U_4(i_4, r_4), i = 1 \text{ batch} \tag{10}$$

We put the core tensor of F'_t that has undergone Tucker decomposition into a stacked deep neural network, providing action as an output. The overall process is in Fig. 3.

3.4. Reinforcement learning

This study solves the portfolio optimization problem. In this case, we rebalance the portfolio at every equal time interval, t days. That means the weight of the portfolio changes after every t^{th} day and becomes asset allocation.

The portfolio has 30 assets, including cash (29 stocks and cash), which is adjusted by changing the weight of 30 assets during each rebalancing period. For time t, the close price of each asset i is assumed to be p_{it}, and the weight of each asset is assumed to be w_{it}.

Here, for all t, the sum of w_{it} is always equal to one. As a vector, the close price vector of all assets at time t is $\overrightarrow{p_t} \in R^{n*1}$, and the weight vector of all assets is $\overrightarrow{w_t} \in R^{n*1}$. Then, the value of the asset at the end of time t is $v_t = \overrightarrow{p_t}^T \bullet \overrightarrow{w_t}$, where T is the transpose and $\bullet$ is the inner product.

The portfolio reward at time t is defined as the logarithmic rate of return of the portfolio value at the end of time t from the portfolio value at time t-1.

$$r_t = \ln v_t / v_{t-1} \tag{11}$$

For total episode L, the sum of all rewards is as follows, where v_L is

Algorithm 1 DDPG algorithm using tucker decomposition with technical indicator and correlation

Input : initial policy parameters θ, Q-function parameters ϕ, replay buffer $\mathcal{D}$
Set target parameters to main parameters $\theta_{target} \leftarrow \theta$,$\phi_{target} \leftarrow \phi$
Observe state s consists of cash and stock holdings, technical indicators and correlations made from stock data. In the initial state, only cash is held.

for episode $= 1, E$ **do**
  Initial random action exploration and observe state s_1
  for all rebalancing time $t = 1, T$ **do**
    select action $a_t = clip(\mu(s_t|\theta) + \mathcal{N}, 0, 1)$ since action weight a_t is the rebalance weight of cash and stock according policy parameter, execute a_t into the environment by buying or selling assets.

    Observe next state s'_t, portfolio return r_t, and done signal d_t to indicate whether or not the entire train period has been learned
    $(s_t, a_t, r_t, s'_t, d_t)$ in replay buffer $\mathcal{D}$
    sample random batch of N transitions (s, a, r, s', d) from $\mathcal{D}$,
    and calculate

$$y = r + \gamma(1 - d)\mathcal{Q}_{\phi_{target}}(s', \mu(s'|\theta_{target}))$$

    In Q-function and policy, the core tensor is extracted by tucker decomposition of the four-dimensional tensor made with the correla tion and technical indicator of each stock, and then it is predicted and used as a feed forward network with θ and ϕ parameters.

    update Q-function and policy parameter θ and ϕ by gradient descent using

$$\nabla \frac{1}{N} \Sigma(\mathcal{Q}_\phi(s, a) - y)^2 \text{ and } \nabla \frac{1}{N} \Sigma(\mathcal{Q}_\phi(s, \mu_\theta(s))$$

    update softly with

$$\phi_{target} \leftarrow \rho\phi_{target} + (1 - \rho)\phi$$
$$\theta_{target} \leftarrow \rho\theta_{target} + (1 - \rho)\theta$$

  end for
end for

Fig. 4. Pseudocode of the proposed method.

the final portfolio value:

$$\sum_{1}^{L} r_t = \sum_{1}^{TL} ln\frac{v_t}{v_{t-1}} = \ln\frac{v_L}{v0}, \mathrm{v}_0 \exp(\sum_{1}^{L} r_t) = v_L \tag{12}$$

Here, reinforcement learning finds a policy network that maximizes v_L. The implementation details are in the pseudocode, Fig. 4.

We consider the fee when calculating the reward to compare it with the actual trading situation. There is no commission when buying, whereas it is 0.3 % when selling. In addition, we assumed rebalancing every ten days.

In the backtest simulation, we considered two assumptions. First, trading by the algorithm does not impact the stock market (zero market impact). Second, due to the high liquidity of all market assets, when placing an order at a close price, the transaction is executed immediately (zero slippage).

3.5. Evaluation

Various metrics are used to measure the performance of a specific portfolio selection strategy. The most reliable information which determines portfolio management's success over time is annualized returns. Annualized returns are an indicator of the extent to which a portfolio returns per year. However, the most significant disadvantage of using these return indicators is that they do not consider the risk associated with the portfolio. Therefore, we considered a risk with the

Table 3
Backtest Result.

Algorithm	SR	MDD	fAPV
1/n	1.6594	10.75	1.6476
Best	1.1217	18.85	1.7473
Close	1.6213	9.52	1.5946
TI	1.6500	9.47	1.5879
TI + cor	1.6094	9.68	1.5764
Jiang, et al. (2017)	1.6097	10.98	1.6870
Yang, et al. (2020)	0.7338	15.45	1.2502
Chen et al. (2019)	1.2794	12.02	1.5027
Wu et al. (2021)	1.6403	9.58	1.5960
Chen et al. (2021)	1.4643	10.56	1.5861
Zero padding	1.6838	9.21	1.6024
Proposed method	**2.0067**	**8.29**	**1.9270**

second metric, that is, Sharpe ratio (Sharpe, 1964). Sharpe ratio is the portfolio's average return to risk and is defined as:

$$\text{Shape Ratio} = \frac{E[\rho_t - \rho_F]}{\sqrt{var(\rho_t - \rho_F)}} \tag{13}$$

where ρ_t is the periodic returns on the proposed portfolio during the test period, and ρ_F is the return on the risk-free asset. We used the US 30-year Treasury bond as a risk-free asset.

The Sharpe ratio (SR) considers the volatility of a portfolio; however, it treats up and down movements in the same manner. We also consider maximum drawdown (MDD) (Magdon-Ismail, Atiya, Pratap, & Abu-Mostafa, 2004) to highlight the decline in our portfolio. MDD refers to the degree of decline of the most significant loss, such that a lower MDD indicates better performance. In a mathematical representation, MDD is defined as,

$$\text{Maximum Drawdown} = \max_{\gamma>t} \frac{p_t - p_\gamma}{p_t} \tag{14}$$

We also use the final accumulative portfolio value (fAPV), which is defined as follows, where v_t is the portfolio value at time t:

$$\text{fAPV} = v_t / v_0 \tag{15}$$

In reinforcement learning, the result is different each time the training is performed. Therefore, to reduce the bias of the results, we trained the model 10 times and reported the average performance measures during the backtest period.

In addition, we compared several algorithms to demonstrate the performance of our algorithm. First, we presented two baselines. The first baseline strategy buys and holds 29 stocks by 1/29 (1/n). The second strategy is to hold only one of the best stocks during the training period (Best). Also, we compared the performance with some modifications of our algorithms in ablation ways. In addition, we compared the performance with existing state-of-the-art algorithms (Jiang, Xu, & Liang, 2017; Yang et al., 2020).

4. Results

4.1. Experimental results

We compared the Sharpe ratio, MDD, and fAPV of the following seven approaches: two baseline algorithms (1/n, Best), a state-of-the-art algorithm with only price history (Close), state-of-the-art algorithm (TI) with DNN, state-of-the-art algorithm with flattened correlation (TI + Corr), and five state-of-the-art algorithms (Chen et al., 2021; Chen et al., 2019; Jiang, et al., 2017; Wu et al., 2021; Yang, Liu, Zhong, & Walid, 2020), the proposed method with only Tucker decomposition (zero padding), and the proposed method. Jiang, et al. (2017) suggested an algorithm called the ensemble of identical independent evaluators (EIIE) topology. Yang, et al. (2020) suggested an ensemble strategy (ES) that automatically selects the best-performing agent among PPO, A2C, and DDPG based on the Sharpe ratio. Chen et al. (2019) and Chen et al. (2021) suggested a novel group genetic algorithm. Wu et al. (2021) suggested an algorithm with two neural networks (CNN and RNN). The results are only in the test period, as described in Table 3. Table 3 shows the algorithm with the highest performance for each evaluation metric in bold.

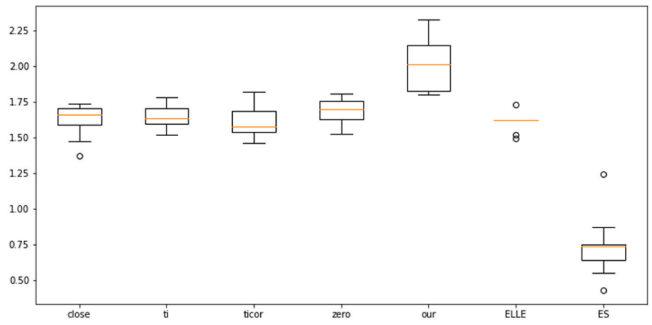

Fig. 5. Box plot of sharpe ratio.

First, 1/n strategy recorded average values for SR, MDD, and fAPV, whereas Best strategy showed high fAPV, but SR and MDD showed inferior performance. It suggests that if one holds only one stock, the risk is high, and portfolio optimization is essential.

Second, Close and TI show similar performances, with TI being slightly better. It suggests that the technical indicator has a strong influence on the results. They also show similar performance in 1/n and SR; however, MDD shows better performance, while fAPV shows the worst performance. This means that the portfolio optimized through the state-of-the-art algorithm is more stable than the 1/n strategy, but the return is low.

Third, TI + cor showed poorer overall performance than TI. It seems that simply using correlation as a feature can have an adverse effect. Fourth, zero padding is a method for Tucker decomposition of the tensor of correlation and technical indicators. Overall, zero-padding shows better performance than the state-of-the-art model. In particular, zero padding shows higher performance than TI + cor, despite using the same features as TI + cor. This result suggests that correlation and technical indicators have different dimensions of information, which can be solved using a tensor-based approach.

Fourth, the state-of-the-art algorithms showed poor performance. Algorithms proposed by Jiang, et al. (2017) showed lower performance in SR and MDD than in TI, but showed better fAPV. In contrast, those proposed by Yang, et al. (2020) showed lower performance in all metrics. Chen et al. (2019) and Chen et al. (2021) also showed lower performance in all metrics. Wu et al. (2021) showed a similar performance to TI but lower performance than Zero padding and the proposed method.

Finally, the performance of the proposed method is superior to that of the other algorithms. It suggests that it is important to perform feature extraction through a 3-dimensional convolutional network and use a tensor rather than simply using the Tucker decomposition.

We reported the model's average value of 10 iterations in Table 3 to avoid any bias in the results. However, the lower bound must show good performance to have practical implications, indicating that only average values should not be compared. Therefore, we compare the distribution of the results of each iteration. Fig. 5 shows the distribution of the Sharpe ratio of each algorithm. Since the lower bound of the proposed method is similar to the upper bound of other algorithms, this result suggests that the proposed method has better performance.

In addition, we compare fAPV as well as the Sharpe ratio. Figs. 6 and 7 show the distribution of the returns in the test period. While other algorithms show a similar trend over time, the proposed method consistently shows the best performance. At the end of the test period, the lower bound of the proposed method performed better than the

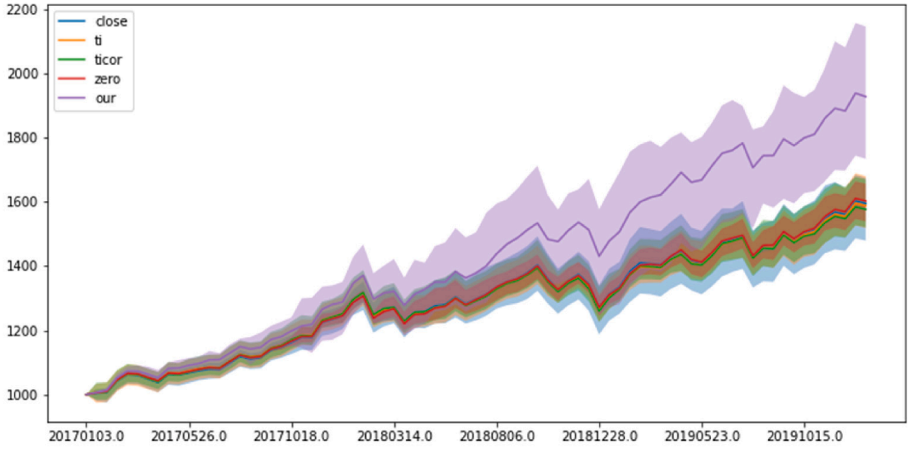

Fig. 6. Portfolio simulation return range comparison.

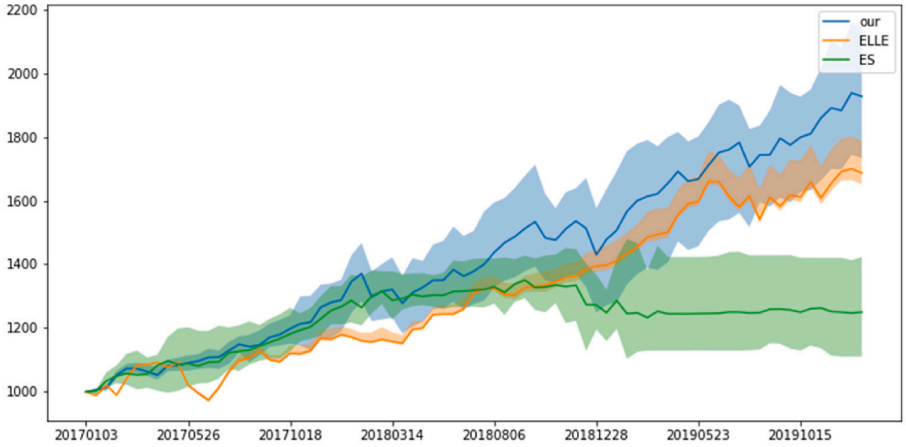

Fig. 7. Portfolio simulation return range comparison.

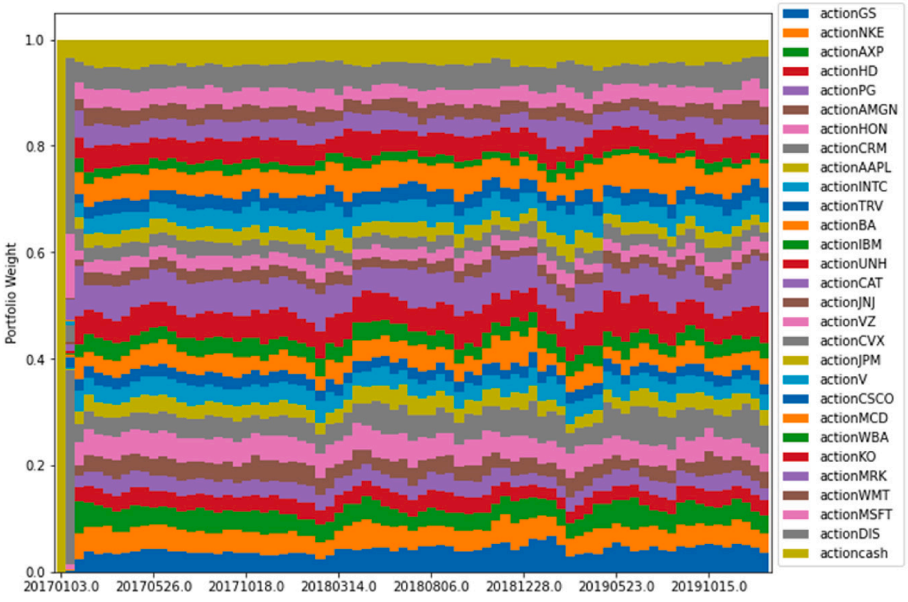

Fig. 8. Action weight of TI.

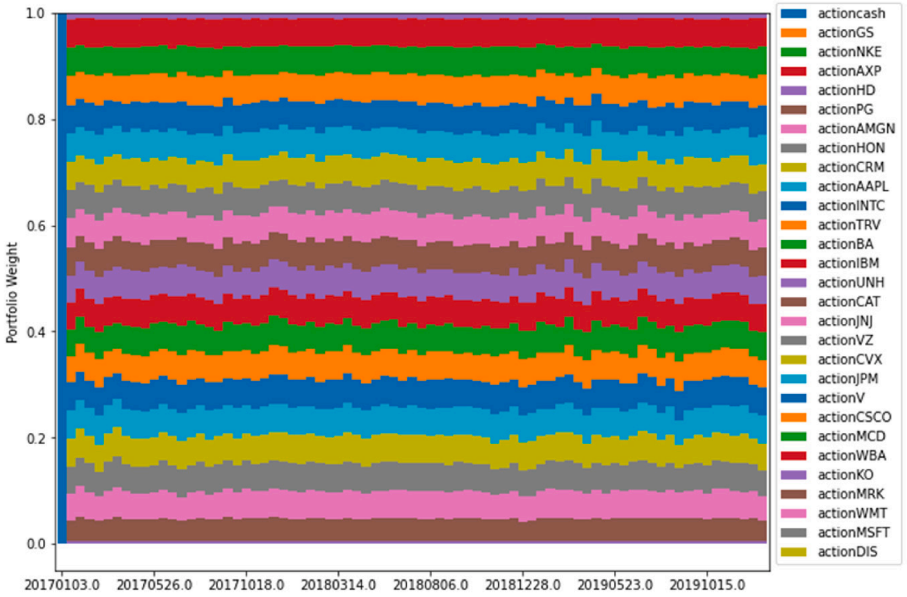

Fig. 9. Action weight of ELLE (Jiang & Liang, 2017).

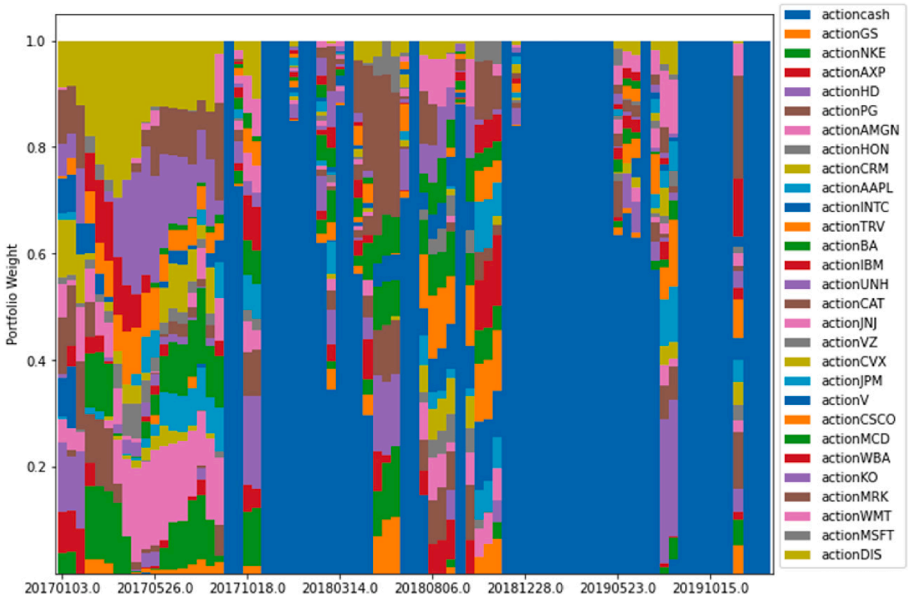

Fig. 10. Action weight of ES (Yang, et al., 2020).

upper bound of the other algorithms. These results validated the performance of the proposed method.

We checked the weight of the actions during the test period to determine the origin of the difference in performance. Fig. 8 shows the action weight trend of TI, and Fig. 8 shows the action weight trend of the proposed method. In Figs. 8 and 9, the sum of weights including weight cash is always one, and the weight of cash is one at the initial point of the test period. In Figs. 8 and 9, TI and ELLE (Jiang, et al., 2017) show almost equal weights to assets and cash in cash at the beginning of the test period. After this beginning period, depending on the market change, the weight change of the assets is not large and is maintained at a similar level.[3] Therefore, it can be concluded that TI performs similarly to the 1/n strategy. In Fig. 10, ES (Yang, et al., 2020) shows the dynamic weight change in the first half of the test period. However, in the second half of the test period, which is a bearish market, ES is passive and mainly holds cash.

However, in Fig. 11, the weight is not divided in the same ratio at the initial point and changes according to the market trend. That is, the proposed method responds sensitively to market flow and reflects it in the model, unlike TI and other algorithms. In addition, the proposed method not only holds cash in a down market but also actively constructs a portfolio.

4.2. Additional results

We also compared the test period by dividing the test period for the robustness check. Table 4 reports the backtest results from January 2017 to June 2018, and Table 5 reports the backtest results from July 2018 to

[3] Although not provided in this paper, algorithms except for zero padding also show similar results.

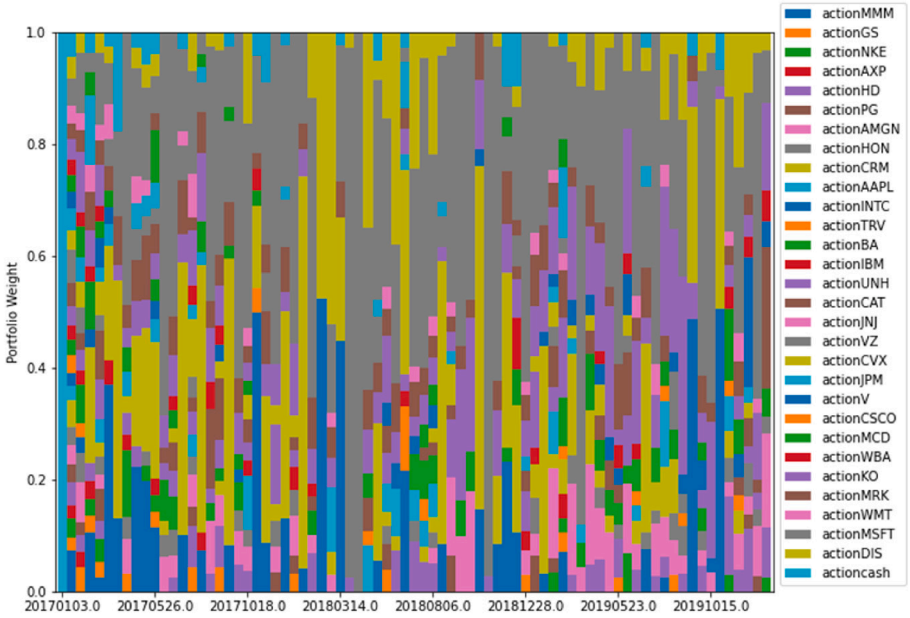

Fig. 11. Action weight of the proposed method.

Table 4
Backtest Result (2017.01 ~ 2018.06).

Algorithm	SR	MDD	fAPV
1/n	1.9884	7.05	1.3118
Best	1.8019	13.6	**1.5278**
Close	1.8948	6.48	1.2824
TI	1.8938	6.95	1.2788
TI + cor	1.8678	6.78	1.2768
(Jiang, et al., 2017)	1.3257	10.98	1.2604
(Yang, et al., 2020)	2.0846	**5.90**	1.3159
Zero padding	1.9032	6.66	1.2791
Proposed method	**2.1092**	7.00	1.3629

Table 5
Backtest Result (2018.07 ~ 2019.12).

Algorithm	SR	MDD	fAPV
1/n	1.3770	10.75	1.2559
Best	0.5613	18.85	1.1433
Close	1.3894	9.52	1.2432
TI	1.4334	9.47	1.2417
TI + cor	1.3801	9.68	1.2346
(Jiang, et al., 2017)	**1.9483**	**7.22**	1.3382
(Yang, et al., 2020)	−0.2688	15.02	0.9500
Zero padding	1.4875	9.21	1.2527
Proposed method	1.9365	8.05	**1.4150**

December 2019. In Table 4, the proposed method shows the highest Sharpe ratio but not the lowest MDD and the highest fAPV. As shown in Table 5, the proposed method showed the best performance in fAPV.

First, the best strategy shows the highest fAPV in Table 4, but the lowest fAPV in Table 5. The Best strategy is to buy the best stock in the training period, which means that the early test period shows good performance because it is similar to the market trend of the training period, but the performance is not maintained over time. Other algorithms, including 1/n, Close, TI, and TI + cor, also produced good results in the first half of the test period, similar to the training period; however, they failed to maintain good results in the second half of the test period because the weights do not change according to the market trend. Algorithms by Jiang, et al. (2017) did not show good performance in the first half, but good performance in the second half. In comparison, algorithms by Yang, et al. (2020) showed good performance in the first half but poor performance in the second half. However, the proposed method adjusts the model weight according to the market trend and consistently produces good performance. That is, the proposed method balances the two state-of-the-art algorithms.

Table 6
Result by TIs (The proposed method).

Timesteps	SR	MDD	fAPV
Close	1.6213	9.52	1.5946
Close, MA	1.7364	9.21	1.7488
Close, MACD	1.8396	8.84	1.7892
Close, RSI	1.8592	9.05	1.8131
Close, MA, MACD, RSI	**2.0067**	**8.29**	**1.9270**

Table 7
Result by the number of days considered in correlation (The proposed method).

TIs	SR	MDD	fAPV
28	1.7158	9.75	1.6685
84	1.8594	9.37	1.8243
126	1.9359	8.81	1.8762
252	**2.0067**	**8.29**	**1.9270**
504	1.9565	8.12	1.9023

We also checked the result by technical indicators. The results are in Table 6. The results show that it is best to include all technical indicators. In particular, MA shows the lowest performance because Close and MA show similar characteristics. In addition, MACD and RSI show similarly good performance, which means that including variables with different characteristics helps increase the model's performance.

We also checked the result by changing the number of days when calculating the correlation. The results are in Table 7. The results show that it is best to consider 252 days when calculating the correlation. In particular, considering a short period when calculating correlation can respond sensitively to the latest trends, but it does not reflect macroscopic movements. Therefore, it shows low performance. In addition, considering an extended period can reflect macroscopic movements, but it does not reflect the latest trends. Therefore, it also shows lower performance than considering one year.

Table 8
Computational Complexities Comparison.

Algorithm	Computational Complexities
Close	O(1*n*m*n)
Jiang, et al. (2017)	O(4*n*m*n)
Yang, et al. (2020)	O(4*n*m*n)
Proposed method	O(R1*R2*R3*R4)

Note: (1) n refers to the number of stocks, and m refers to the number of days to consider (2) R1, R2, R3, and R4 refer to the dimensions of the tensor decomposition core tensor.

4.3. Computational Complexities

Since this study uses a reinforcement learning-based model, it is impossible to compare computing complexity directly. Therefore, we compare the model's policy network parameters in this section, which are in Table 8.

In the proposed method, most of the computational complexity is updating the parameters of the actor/critic network in the DDPG. The proposed method uses a tensor with a size of 4*n*m*n padding using correlation for each technical indicator of each stock as input data. Convert this to a tensor of size 32*n*(m-4)*n using a 3D Convolutional Layer, and reduce the size of the tensor to R1*R2*R3*R4 by tucker decomposition. Therefore, the number of parameters is significantly reduced compared to updating the actor/critic network using the feed-forward network using the 4D tensor.

5. Conclusion

This paper presents an algorithm connecting modern portfolio theory and deep reinforcement learning. We integrated the correlation matrix and technical analysis into a tensor and extracted features from this tensor using a 3D convolutional neural network and Tucker decomposition. We constructed a policy network that predicts this feature using a deep neural network and optimizes the portfolio using DDPG. The results show that the proposed method consistently demonstrates better performance and stability than the other baseline strategies. In addition, the proposed method balances state-of-the-art algorithms and dynamically allocates portfolio assets according to market trends.

This study has the following practical implications: First, we presented an algorithm that dynamically allocates assets according to market trends. When artificial intelligence optimizes the portfolio, risk management is impossible when the market situation changes if the weight is maintained similarly to the initial asset weight. In particular, the proposed method performed well differently from the other algorithms in the downmarket. We overcome this shortcoming and present the possibility of its use in actual trading. Second, we repeated the results ten times and reported the average value of the results. Owing to the nature of reinforcement learning, deterministic results cannot be obtained, and errors in the results may occur. As we iterated ten times and checked the result, we confirmed that the proposed method performs stably and that the lower bound also performs better than state-of-the-art algorithms.

This study has the following academic implications. First, we outperformed state-of-the-art algorithms by combining modern portfolio theory and reinforcement learning. Thus, we emphasize the importance of the traditional approach. Second, we use a multimodal approach to optimize the portfolio. Although many existing studies have solved this problem by using technical analysis, we have suggested a multimodal approach to improve performance. It suggests various research directions, such as portfolio optimization, by combining technical analysis with media effects in the future.

Despite its good results and implications, this study has some limitations. First, we used DDPG for comparison with state-of-the-art algorithms using DDPG. However, with the development of reinforcement learning, many algorithms learn better, and we will use them in future research. Second, we followed the research stream of model development and did not make any reward function adjustments. However, as suggested by the reward function adjustment research stream, adjusting the reward function can produce a stable performance. In future studies, we will study a stable reward function that fits the proposed method.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Aboussalah, A. M., & Lee, C.-G. (2020). Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization. *Expert Systems with Applications, 140*, Article 112891.

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv: 1803.08375*.

Almahdi, S., & Yang, S. Y. (2017). An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications, 87*, 267–279.

Atsalakis, G. S., & Valavanis, K. P. (2009). Surveying stock market forecasting techniques–Part II: Soft computing methods. *Expert Systems with Applications, 36*, 5932–5941.

Ben-Younes, H., Cadene, R., Cord, M., & Thome, N. (2017). Mutan: Multimodal tucker fusion for visual question answering. In *Proceedings of the IEEE international conference on computer vision* (pp. 2612-2620).

Carroll, J. D., & Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika, 35*, 283–319.

Chaouki, A., Hardiman, S., Schmidt, C., Sérié, E., & De Lataillade, J. (2020). Deep deterministic portfolio optimization. *The Journal of Finance and Data Science, 6*, 16–30.

Chen, C.-H., Chen, Y.-H., Diaz, V. G., & Lin, J.-C.-W. (2021). An intelligent trading mechanism based on the group trading strategy portfolio to reduce massive loss by the grouping genetic algorithm. *Electronic Commerce Research*, 1–40.

Chen, C.-H., Chiang, B.-Y., Hong, T.-P., Wang, D.-C., Lin, J.-C.-W., & Gankhuyag, M. (2019). A fuzzy GGA-based approach to speed up the evolutionary process for diverse group stock portfolio optimization. *Journal of Intelligent & Fuzzy Systems, 37*, 7465–7479.

Chen, C.-H., Lu, C.-Y., Hong, T.-P., Lin, J.-C.-W., & Gaeta, M. (2019). An effective approach for the diverse group stock portfolio optimization using grouping genetic algorithm. *IEEE Access, 7*, 155871–155884.

Ding, Q., Wu, S., Sun, H., Guo, J., & Guo, J. (2020). Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction. In *IJCAI* (pp. 4640-4646).

Du, J., Jin, M., Kolm, P. N., Ritter, G., Wang, Y., & Zhang, B. (2020). Deep Reinforcement Learning for Option Replication and Hedging. *The Journal of Financial Data Science, 2*, 44–57.

Elton, E. J., & Gruber, M. J. (1997). Modern portfolio theory, 1950 to date. *Journal of banking & finance, 21*, 1743–1759.

Fernandes, S., Fanaee-T, H., & Gama, J. (2021). Tensor decomposition for analysing time-evolving social networks: An overview. *Artificial Intelligence Review, 54*, 2891–2916.

Fonał, K., & Zdunek, R. (2021). Fast hierarchical tucker decomposition with single-mode preservation and tensor subspace analysis for feature extraction from augmented multimodal data. *Neurocomputing, 445*, 231–243.

Francis, J. C., & Kim, D. (2013). *Modern portfolio theory: Foundations, analysis, and new developments* (Vol. 795). John Wiley & Sons.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., … Osband, I. (2018). Deep q-learning from demonstrations. *In Thirty-second AAAI Conference on Artificial Intelligence*.

Hirshleifer, D. (2015). Behavioral finance. *Annual Review of Financial Economics, 7*, 133–159.

Jiang, Z., & Liang, J. (2017). Cryptocurrency portfolio management with deep reinforcement learning. In *In 2017 Intelligent Systems Conference (IntelliSys)* (pp. 905–913). IEEE.

Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*.

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications, 38*, 5311–5319.

Khemlichi, F., Chougrad, H., Khamlichi, Y. I., El Boushaki, A., & Ali, S. E. B. (2021). Deep Deterministic Policy Gradient for Portfolio Management. In *In 2020 6th IEEE Congress on Information Science and Technology (CiSt)* (pp. 424–429). IEEE.

Kiers, H. A. (2000). Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics: A Journal of the Chemometrics Society, 14*, 105–122.

Li, S., Wu, Y., Cui, X., Dong, H., Fang, F., & Russell, S. (2019). Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 4213-4220).

Lin, S., & Beling, P. A. (2020). An End-to-End Optimal Trade Execution Framework based on Proximal Policy Optimization. In *IJCAI* (pp. 4548-4554).

Lucarelli, G., & Borrotti, M. (2020). A deep Q-learning portfolio management framework for the cryptocurrency market. *Neural Computing and Applications, 32*, 17229–17244.

Magdon-Ismail, M., Atiya, A. F., Pratap, A., & Abu-Mostafa, Y. S. (2004). On the maximum drawdown of a Brownian motion. *Journal of Applied Probability, 41*, 147–161.

Markowitz, H. M. (1952). Portfolio selection. *Journal of Finance, 7*, 77–91.

Minh, D. L., Sadeghi-Niaraki, A., Huy, H. D., Min, K., & Moon, H. (2018). Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. *IEEE Access, 6*, 55392–55404.

Nam, K., & Seong, N. (2019). Financial news-based stock movement prediction using causality analysis of influence in the Korean stock market. *Decision Support Systems, 117*, 100–112.

Park, H., Sim, M. K., & Choi, D. G. (2020). An intelligent financial portfolio trading strategy using deep Q-learning. *Expert Systems with Applications, 158*, Article 113573.

Peng, W., Hong, X., & Zhao, G. (2021). Adaptive Modality Distillation for Separable Multimodal Sentiment Analysis. *IEEE Intelligent Systems*.

Seong, N., & Nam, K. (2021). Predicting stock movements based on financial news with segmentation. *Expert Systems with Applications, 164*, Article 113988.

Seong, N., & Nam, K. (2022). Forecasting price movements of global financial indexes using complex quantitative financial networks. *Knowledge-Based Systems, 235*, Article 107608.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance, 19*, 425–442.

Soleymani, F., & Paquet, E. (2020). Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath. *Expert Systems with Applications, 156*, Article 113456.

Song, Q., Liu, A., & Yang, S. Y. (2017). Stock portfolio selection using learning-to-rank algorithms with news sentiment. *Neurocomputing, 264*, 20–28.

Sun, J., Tao, D., & Faloutsos, C. (2006). Beyond streams and graphs: Dynamic tensor analysis. In *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 374–383).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika, 31*, 279–311.

Vo, N. N., He, X., Liu, S., & Xu, G. (2019). Deep learning for decision making and the optimization of socially responsible investments and portfolio. *Decision Support Systems, 124*, Article 113097.

Wu, M.-E., Syu, J.-H., Lin, J.-C.-W., & Ho, J.-M. (2021). Portfolio management system in equity market neutral using reinforcement learning. *Applied Intelligence, 51*, 8119–8131.

Yang, H., Liu, X.-Y., Zhong, S., & Walid, A. (2020). Deep reinforcement learning for automated stock trading: An ensemble strategy. *Available at SSRN*.

Ye, Y., Pei, H., Wang, B., Chen, P.-Y., Zhu, Y., Xiao, J., & Li, B. (2020). Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, pp. 1112-1119).

Zhang, X., Hu, Y., Xie, K., Wang, S., Ngai, E., & Liu, M. (2014). A causal feature selection algorithm for stock prediction modeling. *Neurocomputing, 142*, 48–59.