IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 December 2023, accepted 23 December 2023, date of publication 26 December 2023,
date of current version 4 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3347608

RESEARCH ARTICLE

Diversified Adaptive Stock Selection Using Continual Graph Learning and Ensemble Approach

JAE-SEUNG KIM, SANG-HO KIM, AND KI-HOON LEE
School of Computer and Information Engineering, Kwangwoon University, Nowon-gu, Seoul 01897, Republic of Korea

Corresponding author: Ki-Hoon Lee (kihoonlee@kw.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant NRF-2022R1F1A1062787; and in part by the Research Grant of Kwangwoon University, in 2022.

ABSTRACT Stock selection is essential for portfolio diversification to reduce risks and maximize profits. However, stock selection is challenging owing to the non-stationary nature of stock markets. In fact, stock markets experience abrupt or gradual concept drift because of their inherent volatility. Concept drift is a phenomenon in which the statistical characteristics of data change over time. Recent stock selection methods have adopted graph neural networks to capture the relational dependencies between stocks. These methods perform non-continual learning that uses a fixed set of stocks without knowledge retention. Non-continual graph learning-based methods can adapt to abrupt concept drift, while continual graph learning-based methods can adapt to gradual concept drift because they involve knowledge retention. To adapt to both abrupt and gradual concept drifts, we propose a stock selection framework called DASS, which combines non-continual and continual models for diversified adaptation. For the both models, we employ graph learning to extract both temporal and relational dependencies. Our graph learning method relies on three main components: 1) low-level temporal modeling, which extracts temporal dependencies of individual stocks; 2) relational modeling, which extracts relational dependencies between stocks; and 3) high-level temporal modeling, which extracts temporal dependencies from the learned relational dependencies. Furthermore, DASS leverages both simple graphs and hypergraphs because they complement each other. The performance of DASS is compared with that of state-of-the-art stock selection methods. Experimental results for stocks included in the Standard & Poor's 500 index reveal that DASS achieves a compounded annual growth rate of 83.2%, outperforming the second-best method by 23.0%P.

INDEX TERMS Stock selection, non-continual learning, continual learning, graph learning, simple graph, hypergraph.

I. INTRODUCTION

A stock portfolio is a combination of different stocks [1], and portfolio diversification is necessary for investors to mitigate risks and improve profits [2]. Stock selection is essential for portfolio diversification, but it is challenging because of the non-stationary nature of stock markets [3]. In fact, stock markets show diverse volatilities, as illustrated in Fig. 1. For instance, phase 4 shows high volatility owing to the COVID-19 (coronavirus disease) pandemic, while phase 3 shows low volatility. Such volatilities may lead to abrupt or gradual concept drift [4], which reflects changes in the statistical properties of data. To consider both abrupt and gradual concept drifts, a diversified approach is necessary rather than relying on an approach to handle either abrupt or gradual concept drift.

Recent studies have shown that deep learning methods can outperform traditional and statistical machine learning methods [5], [6], [7] in the investment field. In particular, recent stock selection methods [8], [9], [10], [11], [12],

The associate editor coordinating the review of this manuscript and approving it for publication was Mu-Yen Chen.

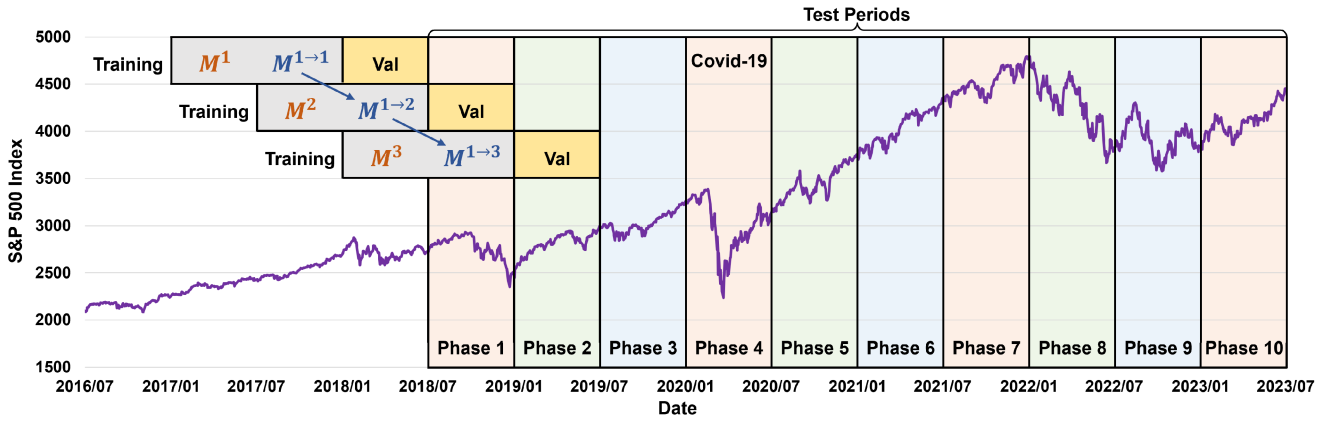

FIGURE 1. Non-continual and continual learning methods in diverse market volatility.

[13], [14], [15], [16], [17], [18], [19] adopt non-continual graph learning, which integrates graph neural networks (GNNs) and non-continual learning on a fixed set of stocks. GNNs are used to capture dependencies between stocks in a graph consisting of a set of nodes (stocks) and edges (relationships). Stock relationships can be constructed using sector-industry, corporate, and price similarity relations.

Among non-continual graph learning-based stock selection methods, those in [8], [12], and [17] employ the walk-forward testing method [20] to account for diverse market volatilities, as illustrated in Fig. 1. This method divides the test data into partitions corresponding to distinct phases. Subsequently, it trains model M^t using a predefined set of stocks and tests the data of phase t. However, the set of stocks can change owing to the non-stationary characteristics of stock markets, including the listing and delisting of stocks. Because non-continual graph learning-based stock selection methods do not consider these changes, an individual model is generated for each phase without reusing models trained in previous phases. Consequently, non-continual graph learning-based stock selection methods may suffer from catastrophic forgetting given the lack of knowledge retention [21] of previously acquired information. The absence of knowledge retention enables adaptation to abrupt concept drift [22], but this adaptation may not be advantageous in markets with low volatility (e.g., phase 3 in Fig. 1).

To mitigate catastrophic forgetting, continual graph learning has been actively investigated in various fields, but it has not yet been widely used in stock selection. Most current continual graph learning-based methods [21], [22], [23], [24], [25], [26] consider knowledge retention by leveraging preceding models. These methods can also be applied to stock selection, as illustrated in Fig. 1. Initially, continual graph learning trains model $M^{1\to1}$ and then tests the data of phase 1. Subsequently, model $M^{1\to t}$ is trained using model $M^{1\to(t-1)}$ and then tested on the data of phase t. Knowledge

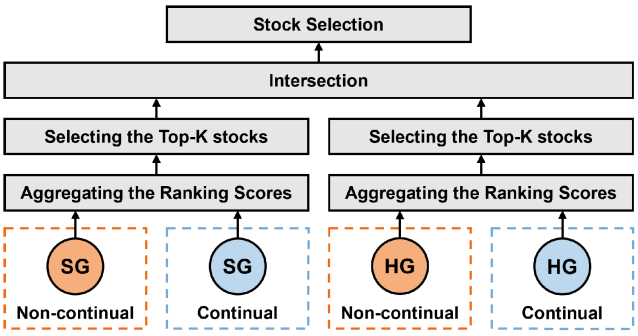

FIGURE 2. Our framework.

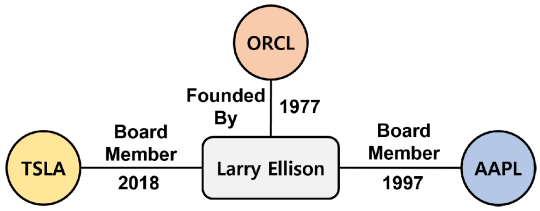

FIGURE 3. An example of relationships [12].

retention enables adaptation to gradual concept drift [22], but this adaptation may not be advantageous in markets with high volatility (e.g., phase 4 in Fig. 1).

In this study, we found that non-continual and continual models are complementary because they adapt to abrupt and gradual concept drifts, respectively. Accordingly, Fig. 2 summarizes our stock selection framework. For diversified adaptation, we aggregate the ranking scores obtained from ranking models in the non-continual and continual models, where a higher ranking corresponds to higher expected return rates [9]. Aggregation is performed separately for both simple graph (SG) and hypergraph (HG). Simple graphs and hypergraphs are used to represent pairwise and collective relationships among stocks, respectively. Fig. 3 illustrates an example of relationships where Oracle (ORCL), Tesla

(TSLA), and Apple (AAPL) are connected via Larry Ellison. In a simple graph, there are three pairwise relationships (i.e., (ORCL, TSLA), (ORCL, AAPL), and (TSLA, AAPL)). In a hypergraph, the three pairwise relationships are represented as a single collective relationship (ORCL, TSLA, and AAPL). Because simple graphs and hypergraphs complement each other [12], we leverage both of them. We select stocks by intersecting the top-K stocks obtained from the aggregated ranking scores for the simple graph and hypergraph.

We propose a diversified adaptive stock selection framework called DASS, which adapts to various market volatilities. First, we combine non-continual and continual models to adapt to abrupt and gradual concept drifts. Second, we extract both temporal and relational information by utilizing low-level temporal, relational, and high-level temporal modeling. Low-level temporal modeling extracts temporal information from time-series features of each stock using a gated recurrent unit (GRU) [27]. Relational modeling extracts relational information from simple graphs and hypergraphs using a graph attention network (GAT) [28] and hypergraph convolution (HConv) [29], respectively. High-level temporal modeling extracts temporal information from the learned relational information using a transformer [30]. Third, we construct simple graphs and hypergraphs based on dynamic time warping (DTW) [31] using stock prices and volume data. Compared with graph learning-based state-of-the-art methods (HATS [8], RSR [9], STH [10], ASA [12], and ALSP [15]), the proposed DASS achieves a compounded annual growth rate (CAGR) of 83.2%, surpassing the second-best method by 23.0%P for stocks included in the Standard & Poor's 500 (S&P500) index, which contains the 500 largest firms in the United States.

The rest of this paper is structured as follows. Section II describes GNNs, the transformer encoder, and continual learning. Section III presents a review of related work. Sections IV and V detail the proposed DASS and present experimental results, respectively. Finally, conclusions and suggestions for future research are described in Section VI. For ease of reading, the abbreviations used in this paper are listed in Table 5 of the Appendix.

II. BACKGROUND

A. TERMINOLOGIES

A simple graph G_s is described by $G_s = (V, E)$, where V is the node set and E is the edge set. The pairwise relationships between the nodes in G_s are represented using a square adjacency matrix. A hypergraph G_h is a generalized graph that captures the collective relationships among nodes. It is described as $G_h = (V, E)$, where V is the node set and E is the hyperedge set. Each hyperedge describes a relationship among a group of nodes. Incidence matrix H of dimensions $N \times M$ is used to connect the nodes to hyperedges, where N and M are the numbers of nodes and hyperedges, respectively. Value $H(p, q)$ is 1 if node p is a member of hyperedge q (i.e., $p \in q$) and 0 otherwise.

B. GNNS

GNNs [32] learn node representations by leveraging inter-node relationships in a simple graph or hypergraph. With the evolution of GNNs, more sophisticated methods have emerged [33], including the graph convolutional network [34] and GAT. The former employs a convolutional neural network to extract local connection patterns, whereas the latter employs a self-attention mechanism [30] to allocate different weights considering the importance of neighboring nodes.

To capture the comparative importance of neighboring nodes around node p (including itself), the GAT allocates weights to the nodes using the attention mechanism. The GAT computes attention coefficient α_{pq} between nodes p and q, as detailed in Equation (1), where h_p^{i-1} and h_q^{i-1} are the representations of nodes p and q at iteration $i-1$, respectively. Furthermore, a and W are learnable parameters, ϕ_1 denotes a non-linear function (e.g., LeakyReLU), and $\|$ denotes concatenation. Attention coefficient α_{pq} reflects the importance of q with respect to p on a simple graph [35].

$$\alpha_{pq} = \frac{\exp(\phi_1(a\left[Wh_p^{i-1} \| Wh_q^{i-1}\right]))}{\sum_{k \in N(p) \cup p} \exp(\phi_1(a\left[Wh_p^{i-1} \| Wh_k^{i-1}\right]))} \tag{1}$$

Using the attention coefficients, the GAT computes a new representation, h_p^i, of node p by taking a weighted sum of neighboring nodes $N(p)$ including p, as expressed in Equation (2), where α_{pq} is the attention coefficient given by Equation (1) and ϕ_2 represents a non-linear activation function (e.g., sigmoid or softmax).

$$h_p^i = \phi_2\Big(\sum_{q \in N(p) \cup p} \alpha_{pq} W h_q^{i-1}\Big) \tag{2}$$

C. TRANSFORMER ENCODER

The transformer [30], which captures temporal dependencies in sequential data, consists of stacked encoder and decoder layers. The encoder layer employs a multi-head self-attention and feed-forward network. To capture the relative importance between all time points in sequential data, multi-head self-attention calculates feature representation R using query Q_h, key K_h, and value V_h, as described in Equation (3). In Equation (3), H is the number of heads and d_f is a scaling factor. Specifically, query $Q_h = EW_h^Q$, key $K_h = EW_h^K$, and value $V_h = EW_h^V$ are linear transformations of temporal embedding E into distinct learnable parameters W_h^Q, W_h^K, and W_h^V, respectively.

$$\begin{aligned} R &= \text{Multihead}(Q_h, K_h, V_h) \\ &= \|_{h=1}^{h=H} \text{Softmax}(Q_h K_h^T / \sqrt{d_f}) V_h \end{aligned} \tag{3}$$

Subsequently, we obtain a new feature representation P via the feed-forward network consisting of two linear layers connected by the activation function ReLU, as expressed by Equation (4), where R is the feature representation given by Equation (3), and W_1, W_2, b_1, and b_2 are

learnable parameters.

$$P = \text{FFN}(R) = \text{ReLU}(RW_1 + b_1)W_2 + b_2 \quad (4)$$

D. CONTINUAL LEARNING

Continual learning is intended to consolidate prior knowledge to prevent catastrophic forgetting. Recent approaches fall into three categories: (1) regularization-based methods [23], [25], which constrain the changes in the model parameters based on their importance, (2) replay-based methods [22], [23], which store a subset of data used in previous tasks, and (3) parameter isolation methods [21], [24], [26], which assign different model parameters to each task.

Elastic weight consolidation (EWC) [36] is a prominent regularization-based method for continual learning. To preserve the model parameters learned in the preceding task, it regulates the model parameters using the Fisher information matrix [37], which represents the importance of the previously learned model parameters. EWC combines loss term $L_{new}(\theta)$ for optimizing the new task and a regularization term, as described by Equation (5), where λ is a hyperparameter that integrates the two loss terms, θ represents the model parameters for the new task, θ^*_{old} represents the optimal model parameters learned for the preceding task, F represents the diagonal elements of the Fisher information matrix, and l is the parameter index.

$$L(\theta) = L_{new}(\theta) + \frac{\lambda}{2}\sum_l F_l(\theta_l - \theta^*_{old,l})^2 \quad (5)$$

III. RELATED WORK

Our study is related to recent work based on non-continual graph learning, continual graph learning, and transformers in various fields, including investments.

A. NON-CONTINUAL GRAPH LEARNING-BASED METHODS

Non-continual graph learning-based methods [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19] that capture inter-stock relational dependencies have been proposed for stock selection. Kim et al. [8] introduced a hierarchical GAT called HATS to consolidate information from neighboring nodes and various relationship types. Feng et al. [9] proposed a relational stock ranking framework called RSR, which incorporates temporal graph convolutions to capture time-sensitive relationships between stocks. Sawhney et al. [10] introduced a spatio-temporal hypergraph attention network called STH, which integrates spatial hypergraph convolutions and the Hawkes process with attention mechanism to consider spatial and temporal dependencies. Hsu et al. [11] proposed a financial GAT to recommend stocks using hierarchical relationships between stocks and sectors. Kim et al. [12] proposed a portfolio management framework called ASA that integrates ranking models with classification and regression models for selecting stocks and determining the investment ratio. Feng et al. [13] introduced a relation-aware dynamic attributed GAT to capture local correlation topology information. He et al. [14] proposed a static-dynamic GNN to consider potential relationships between stocks. Heyuan et al. [15] presented an adaptive long-short pattern transformer called ALSP to distill stock patterns at different context scales. Ma et al. [16] devised an attribute-driven fuzzy hypergraph network to quantify the intensity of collective relationships among stocks and simulate their influence. Huynh et al. [17] proposed a profit-driven framework to capture non-pairwise correlations and individual stock patterns. Song et al. [18] proposed a multi-relational graph attention ranking network to capture dependencies between stocks from industry, Wiki, and price similarity relations. Tian et al. [19] proposed a graph evolution recurrent unit that directly learns dynamic temporal dependencies between stocks from time-series data.

Although changes in the set of edges have been considered in [13], [14], [15], [16], [18], and [19], changes in the set of nodes and knowledge retention have been neglected.

B. CONTINUAL GRAPH LEARNING-BASED METHODS

Continual graph learning-based methods [21], [22], [23], [24], [25], [26] that address catastrophic forgetting have been proposed in various fields. Wang et al. [23] proposed a streaming GNN model to capture new patterns and integrate existing information. Perini et al. [22] proposed a streaming GNN model using experience replay to select rehearsal samples and update the network parameters incrementally. He et al. [24] proposed a model isolation-based continual graph learning method to capture changes in user preferences. Tan et al. [25] proposed a heterogeneous continual GNN to learn temporal and spatial dependencies in future prices. Zhang et al. [21] proposed a parameter isolation GNN to continually learn new patterns while freezing the parameters that learn stable patterns. Wang et al. [26] proposed a continual learning framework that combines an influence-based knowledge expansion strategy and memory-augmented knowledge consolidation mechanism to expand new knowledge and preserve old knowledge. Despite the major developments in continual graph learning, its application to stock selection has remained largely unexplored.

C. TRANSFORMER-BASED METHODS

Transformer-based methods [15], [38] that capture temporal dependencies in sequential data have been proposed in various fields. Jina et al. [38] proposed a time-series anomaly detection method that employs a stacked transformer encoder and one-dimensional convolutional neural network (1D-CNN) [39]-based decoder, to capture global trends and local variability in time-series data. ALSP [15] is a stock selection method that uses a transformer encoder to capture long- and short-term stock patterns. In this paper, we propose a sophisticated decoder suitable for stock selection to capture local and global information.

IV. PROPOSED STOCK SELECTION FRAMEWORK

We propose a stock selection framework called DASS that employs continual graph learning and an ensemble approach for diversified adaptation.

A. PROBLEM FORMULATION

To reduce the disparity between stock movement predictions and investment returns, we use a ranking approach for stock selection [9]. In a given set of N stocks, each stock involves feature vector $X_i = [x_{t-\Delta T}, \ldots, x_{t-1}]$ on trading day t, where ΔT is the size of a sliding window. The output of the ranking model is an ordering of the stocks, $Y_t = \{y_t^1 > y_t^2 \cdots > y_t^N\}$, with respect to expected return rates on day t.

B. ARCHITECTURE

Fig. 4 shows the architecture of DASS, which achieves diversified adaptation. The proposed architecture preprocesses stock data and constructs simple graphs and hypergraphs. Subsequently, a graph learning method is applied to both the non-continual and continual models to capture temporal and relational dependencies. Finally, stocks are selected by combining the non-continual and continual models through an ensemble approach. This architecture enables a synergistic interaction in which the non-continual model, which adapts to abrupt concept drift, and the continual model, which adapts to gradual concept drift, complement each other. Furthermore, continual graph learning enables the continual model to adapt to changes in node and edge sets with knowledge retention.

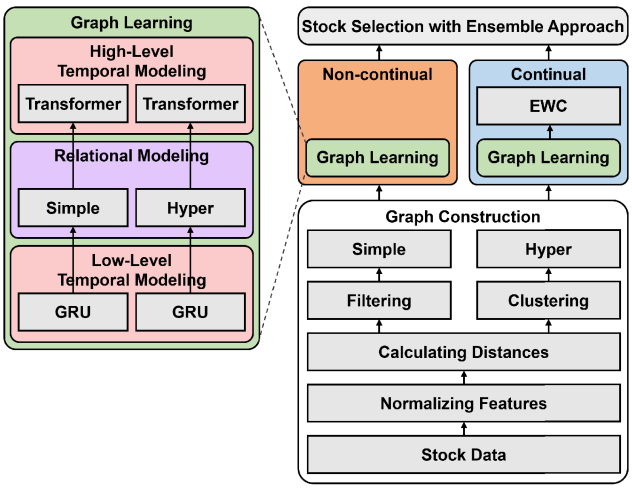

FIGURE 4. Architecture of DASS.

C. GRAPH CONSTRUCTION

We use only stock information, including the opening, highest, lowest, closing, and volume values. The collected stock data are normalized using MinMax scaling. We calculate the distance between each pair of stocks using multi-dimensional DTW [40], which determines the minimum alignment cost between two sequences. Subsequently, we construct simple graphs and hypergraphs based on the distance. For the simple graph, we filter out trivial edges to reduce noise by sorting edges based on the distance and attaching only their lower $\epsilon\%$, where ϵ is a hyperparameter. For the hypergraph, we cluster the target stock with the nearest stocks in terms of the distance to generate hyperedges (i.e., collective relationships). To cluster stocks, we employ the K-nearest neighbor algorithm [41], where K is a hyperparameter that determines the number of stocks included in the hyperedge.

D. GRAPH LEARNING

A graph learning method is employed to capture both temporal and relational information. This method comprises (1) low-level temporal, (2) relational, and (3) high-level temporal modeling. Low-level temporal modeling captures temporal information of individual stocks. Relational modeling captures relational information between stocks in both simple graphs and hypergraphs. High-level temporal modeling further captures temporal information from the learned relational information. The following subsections detail these graph learning components.

1) LOW-LEVEL TEMPORAL MODELING

A sliding window of the normalized feature vectors is used as the input for low-level temporal modeling. To capture the temporal dependencies of individual stocks, we use one GRU layer, as depicted in Fig. 4.

2) RELATIONAL MODELING

For relational modeling, we leverage both simple graphs and hypergraphs because the approach of using them together showed good performance in recent study [12]. To capture relational features, we use different networks for the simple graph and hypergraph, as depicted in Fig. 5. For the simple graph, we leverage the GAT to assign different weights to neighboring nodes. We employ two GAT layers with the activation functions ELU and LeakyReLU. For the hypergraph, we leverage HConv with a multi-head attention mechanism [29]. We employ two HConv layers with the activation functions ELU and LeakyReLU.

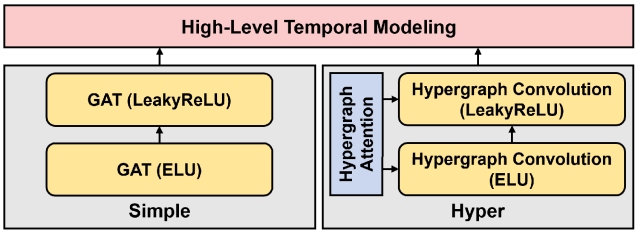

FIGURE 5. Architecture of relational modeling.

3) HIGH-LEVEL TEMPORAL MODELING

For high-level temporal modeling, we leverage a transformer encoder because the approach of using it in stock selection showed superior performance over other methods [15]. Fig. 6 shows the architecture of high-level temporal modeling. As the encoder, we employ a stacked transformer to enhance

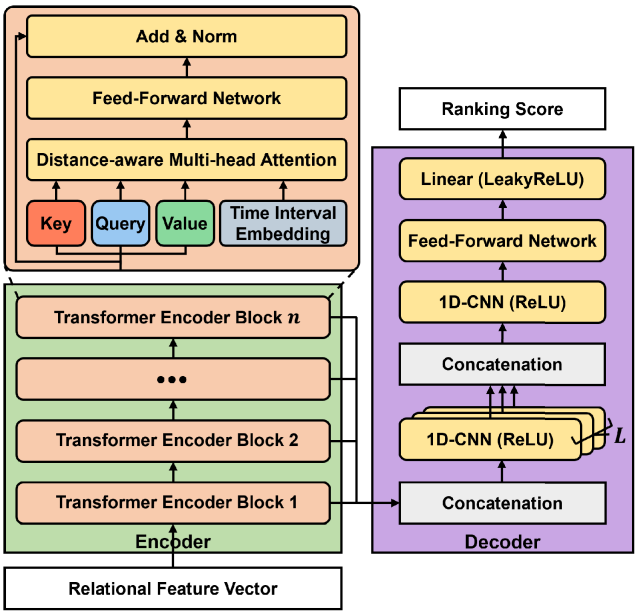

FIGURE 6. Architecture of high-level temporal modeling.

the feature representation of the relational feature vector. To fully exploit the potential information of preceding layers, we concatenate the outputs of all the transformer encoder blocks and use them as the input to the decoder.

Each transformer encoder block employs a time interval embedding [42] to capture the temporal distance between all time points. It first calculates feature representation $\bar{R}$, as expressed in Equation (6), where Q_h, K_h, and V_h are the query, key, and value, respectively, d_f is the scaling factor, Z_h is the matrix that represents the temporal distance between all time points, and H is the number of heads. Subsequently, we apply the feed-forward network defined in Equation (4) to $\bar{R}$ and a residual connection between the input of the transformer encoder block and output of the feed-forward network.

$$\bar{R} = ||_{h=1}^{h=H} \text{Softmax}\left(\frac{\text{ReLU}(Q_h K_h) * Z_h}{\sqrt{d_f}}\right) V_h \tag{6}$$

For the decoder, we adopt a hierarchical structure of 1D-CNNs to extract local and global information from a multi-level representation obtained from the encoder. To extract local information, we use multiple 1D-CNNs with different kernel sizes to capture the dependencies across long and short time scales. After concatenating the outputs of all the 1D-CNNs, we employ another 1D-CNN to extract global information. Finally, we apply the feed-forward network defined in Equation (4) and then predict the ranking score using a linear layer with the activation function LeakyReLU.

E. STOCK SELECTION WITH ENSEMBLE APPROACH

Algorithm 1 describes the diversified adaptive stock selection algorithm. For diversified adaptation, we integrate the ranking models in non-continual and continual models for simple graph and hypergraph. Specifically, we combine the ranking scores obtained from simple graph-based ranking models (SG_{NC} and SG_C) and hypergraph-based ranking models (HG_{NC} and HG_C) using an *AGGREGATE* function (lines 1 and 2). We use averaging as the *AGGREGATE* function. Subsequently, we intersect the top-K stocks obtained from aggregated ranking scores R_{SG} and R_{HG} to select the stocks with the highest expected profits (line 3).

Algorithm 1 Diversified Adaptive Stock Selection Algorithm

Input: (1) the ranking models SG_{NC} and HG_{NC} in the non-continual model,
(2) the ranking models SG_C and HG_C in the continual model

Output: the selected stock set $S_{selected}$

1. Aggregate the ranking scores of the simple graph-based ranking models
   $R_{SG} \leftarrow AGGREGATE(SG_{NC}, SG_C)$
2. Aggregate the ranking scores of the hypergraph-based ranking models
   $R_{HG} \leftarrow AGGREGATE(HG_{NC}, HG_C)$
3. $S_{selected} \leftarrow topK(R_{SG}) \cap topK(R_{HG})$
4. **return** $S_{selected}$

To optimize ranking models SG_{NC} and HG_{NC} in the non-continual model, we employ a loss function [15] that combines ranking loss L_R and graph proximity loss L_{GP}, as described in Equation (7), where η is a hyperparameter to combine the two loss terms. Ranking loss L_R integrates the pointwise regression loss and pairwise ranking-aware loss, as described in Equation (8), where r and $\hat{r}$ represent the ground-truth and predicted ranking vectors, respectively, N is the number of stocks, and ρ is a hyperparameter that combines the two loss terms. Graph proximity loss $L_{GP}(i)$ regulates the representation of node i by gathering its neighboring nodes together and pushing its non-neighboring nodes farther. This is described by Equation (9), where $N(i)$ is a set of neighboring nodes of node i, S is the set of all the stocks, and o_i, o_j, and o_p denote the representations of node i, its neighbor node j, and a sampled non-neighbor node p, respectively.

$$L_{NC} = L_R + \eta L_{GP} \tag{7}$$

$$L_R = \|\hat{r} - r\|^2 + \rho \sum_{i=0}^{N} \sum_{j=0}^{N} \max(0, -(\hat{r}_i - \hat{r}_j)(r_i - r_j)) \tag{8}$$

$$L_{GP}(i) = -\sum_{j \in N(i) \cup i} \log(\sigma(o_i o_j)) - \sum_{p \in S - N(i) \cup i} \log(\sigma(-o_i o_p)) \tag{9}$$

To optimize ranking models SG_C and HG_C in the continual model, we employ the EWC loss function, which integrates loss function $L_{new}(\theta)$ of the new task and the regularization term defined in Equation (5), where we use L_{NC} defined in Equation (7) as $L_{new}(\theta)$.

F. ANALYSIS OF STOCK SELECTION METHODS

Table 1 compares the proposed DASS framework with graph learning-based state-of-the-art stock selection methods in terms of temporal modeling, relational modeling, and learning approach. ALSP [15] does not use an RNN for temporal modeling or hypergraphs for relational modeling, and it does not consider continual graph learning. ASA [12] uses neither a transformer for temporal modeling nor continual learning. RSR [9] does not use a transformer for temporal modeling, hypergraph for relational modeling, or continual graph learning. STH [10] does not use a transformer for temporal modeling, simple graphs for relational modeling, or continual learning. HATS [8] does not use a transformer for temporal modeling or hypergraphs for relational modeling, and it considers neither ranking nor continual graph learning. To the best of our knowledge, DASS is the only method that combines non-continual and continual models to adapt diverse market volatilities. Moreover, DASS is a comprehensive solution that effectively integrates various techniques (i.e., GRU, transformer, simple graph, hypergraph, ranking approach, and continual graph learning) for the temporal modeling, relational modeling, and learning approach dimensions. Although DASS integrates some techniques used in existing methods, the combination of these techniques to yield positive results is not straightforward.

TABLE 1. Comparison of stock selection methods.

	Temporal Modeling		Relational Modeling		Learning Approach	
	RNN	transformer	simple graph	hypergraph	ranking	continual
DASS	GRU	○	○	○	○	○
ALSP [15]	×	○	○	×	○	×
ASA [12]	LSTM	×	○	○	○	×
RSR [9]	LSTM	×	○	×	○	×
STH [10]	LSTM	×	×	○	○	×
HATS [8]	LSTM	×	○	×	×	×

V. EXPERIMENTS

A. EXPERIMENTAL SETUP

1) DATASETS

We evaluated the proposed DASS framework using stocks included in the S&P500 index. We collected stock data from Yahoo Finance [43], including the opening, highest, lowest, closing, and volume values. To consider diverse market conditions, we tested non-continual and continual models using the walk-forward testing method, as illustrated in Fig. 1. On average, the periods of training, validation, and testing per phase were 252 days, 126 days, and 126 days, respectively. The validation data were used to select the model with the lowest validation loss. For the non-continual and continual models, simple graphs and hypergraphs were generated on the first day of the validation period per phase using stocks that remained within the S&P 500 index throughout the validation and test periods.

To simulate real-world trading conditions, we started with an initial balance of $10,000, only took long positions, and assumed a transaction cost rate of 0.1%, unless stated otherwise. For the sake of simplicity, we purchased the maximum available number of shares per stock and sold them at the closing price the following day.

2) EVALUATION METRICS

The return rate and risk indicators (i.e., Sharpe ratio—SR [44] and maximum drawdown—MDD [45]) were used to evaluate the performance of each method. The return rate was computed as the ratio of portfolio value PV_{end} after the test period to initial balance PV_{start} as $(PV_{end} - PV_{start})/PV_{start}$.

The SR evaluates the return of an investment relative to its risk [46], as defined in Equation (10), where $\mathbb{E}[R]$ denotes the expected return and $\sigma[R]$ denotes the standard deviation of return, which quantifies the fluctuation (i.e., risk). A higher SR indicates a higher risk-adjusted return. In Equation (10), we use the portfolio change rate as the return.

$$\text{SR} = \frac{\mathbb{E}[R]}{\sigma[R]} \tag{10}$$

The MDD evaluates the maximum loss rate in a portfolio from its peak to its trough over a specified period T, as expressed by Equation (11). The inner maximum term calculates the drawdown for time τ.

$$\text{MDD}(T) = \max_{\tau \in (0,T)} \left[\max_{t \in (0,\tau)} \frac{PV_t - PV_\tau}{PV_t} \right] \tag{11}$$

3) COMPARISON METHODS

The proposed DASS method was compared with the graph learning-based state-of-the-art methods described below. We evaluated the performance across the top-5, 10, and 15 stocks and recorded the best performance per method. We employed the Adam optimizer [47] with momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, epsilon of 10^{-7}, decay of 0.99, mini-batch size of 8, and learning rate of 0.001. Knowledge-based simple graphs and hypergraphs were constructed according to the graph construction method of ASA [12] for optimal performance. For each method, we optimized the network architectures and hyperparameters as listed below. The others were set as indicated in the original studies.

- In the **buy and hold (B&H)** method, the S&P500 index is bought at the start of each test phase. Then, it is held and subsequently sold at the end of that phase.
- **HATS [8]** leverages long short-term memory (LSTM) and hierarchical attention mechanism. The number of LSTM units was set to 64, the dropout rate was set to 0.5, and the number of top-K stocks was set to 5.
- **STH [10]** leverages the Hawkes process with attention mechanism and hypergraph convolution using a

TABLE 2. Experimental results for each phase.

Phases	Return rate							Sharpe ratio (SR)							Maximum drawdown (MDD)						
	B&H	HATS	STH	RSR	ASA	ALSP	DASS	B&H	HATS	STH	RSR	ASA	ALSP	DASS	B&H	HATS	STH	RSR	ASA	ALSP	DASS
1	-8.9%	-9.2%	-10.9%	-7.4%	11.6%	5.8%	**13.8%**	-0.93	-0.99	-0.75	-0.36	0.82	0.79	**1.02**	0.20	0.22	0.21	0.21	**0.07**	0.17	0.09
2	17.1%	20.6%	19.7%	10.7%	31.8%	33.4%	**35.9%**	1.49	1.85	1.88	0.69	2.04	2.26	**2.31**	**0.07**	0.10	0.12	0.18	0.08	0.16	0.14
3	8.3%	9.7%	10.5%	20.9%	24.5%	38.7%	**40.4%**	0.60	0.75	0.77	1.26	1.31	1.72	**1.97**	**0.06**	0.09	0.14	0.09	0.11	0.12	0.08
4	-5.1%	-8.2%	-6.5%	69.1%	12.8%	**78.6%**	62.9%	-0.11	-0.15	-0.14	1.28	0.44	**1.52**	1.35	0.34	0.38	0.37	0.35	0.36	0.32	**0.31**
5	19.1%	24.8%	42.8%	26.6%	64.6%	51.7%	**68.4%**	1.29	1.44	1.48	1.62	1.63	1.59	**1.74**	**0.10**	0.11	0.14	0.11	0.16	0.15	0.14
6	16.0%	21.3%	33.1%	25.6%	**49.0%**	32.5%	40.6%	1.31	1.39	1.76	1.60	2.30	1.84	**2.31**	**0.04**	0.07	0.11	0.10	0.06	0.09	0.08
7	9.8%	3.4%	14.9%	1.3%	24.4%	20.1%	**25.3%**	0.74	0.61	0.77	0.26	0.79	0.78	**0.84**	**0.05**	0.08	0.13	0.16	0.11	0.12	0.09
8	-21.0%	-17.5%	-38.2%	-33.3%	6.2%	-23.6%	**10.4%**	-1.43	-1.08	-2.57	-1.06	1.16	-0.97	**1.28**	0.23	0.22	0.39	0.38	**0.05**	0.35	0.08
9	1.3%	6.6%	19.1%	9.4%	11.9%	24.8%	**27.5%**	0.08	0.65	1.03	0.71	0.83	1.09	**1.23**	0.17	**0.10**	0.13	0.19	0.16	0.14	0.11
10	16.3%	12.5%	22.3%	16.7%	18.1%	31.3%	**39.6%**	**1.51**	0.89	1.12	1.02	1.05	1.15	1.27	**0.08**	0.14	0.21	0.15	0.19	0.20	0.21
Min.	-21.0%	-17.5%	-38.2%	-33.3%	6.2%	-23.6%	**10.4%**	-1.43	-1.08	-2.57	-1.06	0.44	-0.97	**0.84**	**0.04**	0.07	0.11	0.09	0.05	0.09	0.08
Max.	19.1%	24.8%	42.8%	69.1%	64.6%	**78.6%**	68.4%	1.51	1.85	1.88	1.62	2.30	2.26	**2.31**	0.34	0.38	0.39	0.38	0.36	0.35	**0.31**
Avg.	5.3%	6.4%	10.7%	14.0%	25.5%	29.3%	**36.5%**	0.46	0.54	0.54	0.70	1.24	1.18	**1.53**	0.13	0.15	0.20	0.19	0.14	0.18	**0.13**
Std.	**0.13**	0.14	0.24	0.26	0.19	0.27	0.19	1.04	0.99	1.36	0.87	0.59	0.89	**0.52**	0.10	0.10	0.10	0.10	0.09	0.09	**0.07**

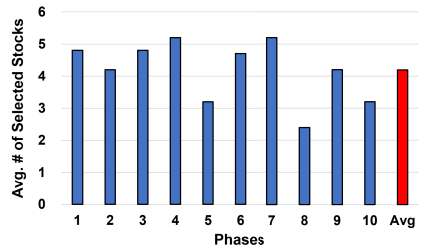

FIGURE 7. Average number of selected stocks of DASS for each phase.

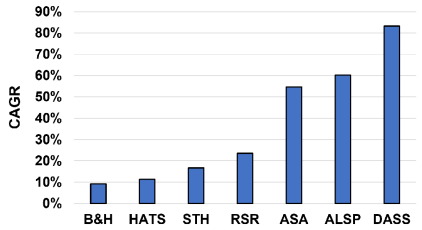

FIGURE 8. Comparison with other methods.

multi-head attention mechanism. The sliding window size was set to 16 and the number of top-K stocks was set to 5.

- **RSR** [9] leverages LSTM and temporal graph convolution. The number of LSTM units was set to 64, the dropout rate was set to 0.5, and the number of top-K stocks was set to 5.
- **ASA** [12] leverages the Hawkes process with attention mechanism, hierarchical attention mechanism, and hypergraph convolution using a multi-head attention mechanism.
- **ALSP** [15] leverages the hierarchical transformer and time-adaptive modulator. The graph sparsity was set to 0.9, the sliding window size was set to 16, and the number of top-K stocks was set to 5.
- **ALSP+C** is an enhanced ALSP method that verifies the effectiveness of using only continual graph learning. The number of top-K stocks was set to 5.
- **ALSP+E** is an enhanced ALSP method that applies the proposed ensemble approach to diversified adaptation, as in Algorithm 1, by using non-continual and continual models. The number of top-K stocks was set to 5.

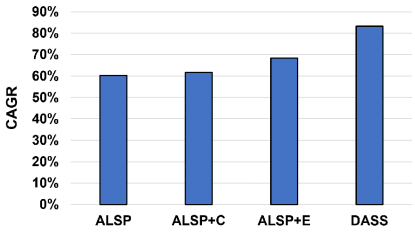

FIGURE 9. Comparison with stock selection method based on continual graph learning.

4) IMPLEMENTATION DETAILS OF DASS

For graph construction, we set filtering threshold ϵ for the simple graph to 10% and clustering threshold K for the hypergraph to 5. For low-level temporal modeling, we set the size of the sliding window, ΔT, to 16 and number of GRU units to 32. For relational modeling, we set the number of heads for the GAT layer to 1 and number of channels for the HConv layer to 32. For high-level temporal modeling, we set scaling factor d_f to 16, number H of heads to 6, number n of transformer encoder blocks to 3, number L of 1D-CNNs that extract local information to 3, where each kernel size is 2, 4, and 8, and the kernel size of the 1D-CNN that extracts global information to 2. In stock selection using the ensemble approach, we set K in the top-K selected stocks to 10, λ in Equation (5) to 0.5, η in Equation (7) to 0.5, and ρ in Equation (8) to 4. Furthermore, we set the number of epochs for training to 10.

B. EXPERIMENTAL RESULTS

1) COMPARISON WITH OTHER METHODS

As demonstrated in Table 2, DASS outperformed the comparison methods in all phases, except for phases 4 and 6, in terms of return rate. DASS achieved an average return rate of 36.5%, outperforming the second-best method, ALSP, by 7.2%P. Specifically, in phases 4 and 8, which had higher volatility than the preceding phase, DASS exhibited less fluctuation in the return rate than the other methods. For phase 8 with a downward trend, all the methods incurred substantial

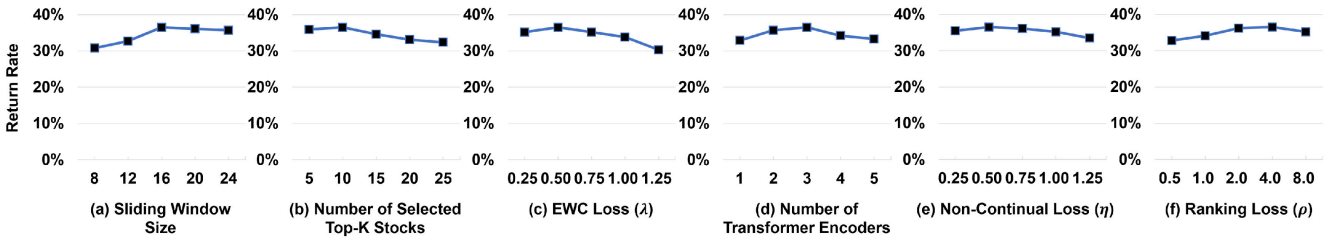

FIGURE 10. Comparison with other hyperparameter values.

TABLE 3. Number of days of investment during the whole test period.

Methods	Profit days	Loss days	No trade	Total
DASS	666 (53%)	528 (42%)	64 (5%)	1258
ALSP	685 (54%)	573 (46%)	0 (0%)	1258

losses, except for ASA and DASS, which made a good profit. These results occurred because DASS selected a varying number of stocks based on the volatilities of the market by intersecting the top-K stocks acquired from the two ranking scores, R_{SG} and R_{HG}, which integrate the non-continual and continual models, as in Algorithm 1. As shown in Fig. 7, DASS selected, on average, fewer than three stocks per day in phase 8. This was because there were only a limited number of profitable stocks when the market showed a downward trend.

Regarding risk indicators, the average SR of DASS was 1.53, outperforming the second-best method, ASA, by 0.29. The average MDD of DASS was 0.13, outperforming the second-best method, B&H, by 0.001. In phases 4 and 6, DASS exhibited the third- and second-best return rates, and ALSP and ASA were the best methods, respectively. This was because ALSP and ASA prioritized maximizing profits rather than minimizing risks, as indicated by the average performance of the risk indicators SR and MDD. Compared with ALSP and ASA, DASS achieved a harmonious balance between profit and risk by incorporating various ranking models in non-continual and continual models.

Fig. 8 shows the CAGR for each method over the whole test period. DASS outperformed the comparison methods, achieving a CAGR of 83.2%, which was 23.0%P higher than that of the second-best method, ALSP. This was because DASS yielded higher profits even when the market showed high volatility in phases 1, 4, and 8, leading to a compound interest effect. As listed in Table 3, DASS achieved 4%P fewer days of losses compared with the second-best method, thus mitigating the adverse effects of losses on compounding.

Overall, the evaluation results indicate that DASS, which integrates non-continual and continual models and incorporates all the techniques listed in Table 1, is effective for stock selection.

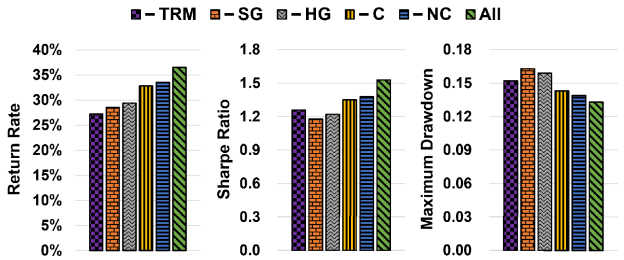

FIGURE 11. Experimental results of the ablation studies.

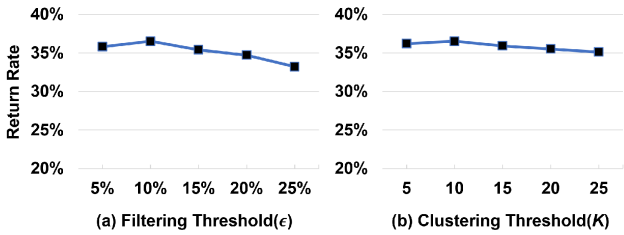

FIGURE 12. Comparison with hyperparameters for graph construction.

2) COMPARISON WITH STOCK SELECTION METHOD BASED ON CONTINUAL GRAPH LEARNING

Fig. 9 shows the effects of applying continual graph learning to the second-best method (ALSP). ALSP+C achieved a higher CAGR than ALSP, but the improvement was small (1.4%P). In contrast, ALSP+E exhibited a substantial improvement over ALSP (8.2%P). These results demonstrate that our ensemble approach can maximize profits through diversified adaptive stock selection. Moreover, DASS exhibited a 14.8%P higher performance than ALSP+E, confirming the effectiveness of our framework.

3) ABLATION STUDIES

Ablation studies were conducted to evaluate the contribution of each component used in DASS. The transformer (TRM), simple graph (SG), hypergraph (HG), continual model (C), and non-continual model (NC) were excluded. "ALL" indicates DASS with all the components. Fig. 11 shows the average performance of the DASS variant models. The results indicate that all the components contributed to the framework performance. In particular, the average

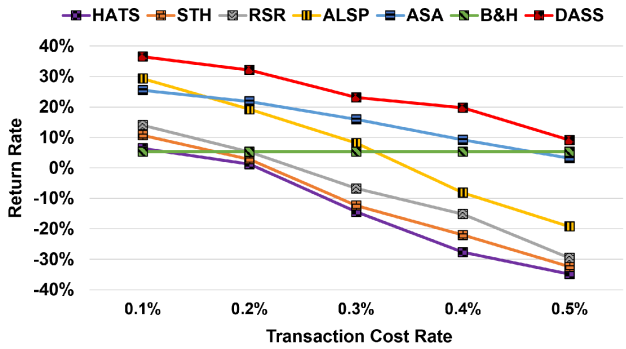

FIGURE 13. Experimental results of the robustness study.

performance was degraded when either the continual model (C) or non-continual model (NC) was used exclusively. These results confirm the complementarity between the components of DASS.

4) COMPARISON WITH HYPERPARAMETERS FOR GRAPH CONSTRUCTION

Fig. 12 shows the performance of DASS according to the hyperparameters for constructing the simple graph and hypergraph (i.e., the filtering and clustering thresholds). As shown in Fig. 12(a), the performance was degraded as the filtering threshold increased because many trivial edges were included in the simple graph. Fig. 12(b) indicates that the performance was degraded as the clustering threshold increased because many dissimilar stocks were included in the hyperedge.

5) COMPARISON WITH OTHER HYPERPARAMETER VALUES

Fig. 10 shows the performance of DASS according to various hyperparameters including the sliding window size for ranking models, K in the top-K stocks selected from each ranking model, number of transformer encoders, λ in EWC loss (Equation (5)), η in the non-continual loss (Equation (7)), and ρ in the ranking loss (Equation (8)). As shown in Fig. 10(a), if the sliding window size was very small or large, the performance was degraded because of insufficient or excessive information. Fig. 10(b) shows that the performance was degraded as K increased because low-profit stocks were included in the portfolio for high K values. Fig. 10(c) shows that the performance was degraded as λ increased because the continual model was adapted too slowly to the market volatilities. Figs. 10(d) to (f) indicate that the performance was not very sensitive to the number of transformer encoders, η, and ρ. Furthermore, we also verified that the performance was insensitive to other hyperparameters (e.g., learning rate, the number of epochs, and mini-batch size).

6) ROBUSTNESS STUDY

Transaction costs (e.g., transaction fees and taxes) are important factors in stock trading. Hence, we evaluated the influence of the transaction cost rate on the DASS performance to confirm its robustness. Fig. 13 shows the average return rate for varying transaction cost rates. The average return rate decreased with increasing transaction cost rate for all the methods. Nevertheless, DASS exhibited superior performance even when the transaction cost rate reached 0.5%, which exceeds typical real-world trading costs. This performance can be attributed to DASS generating more profit per trade, overcoming the related transaction cost.

TABLE 4. Average running time per phase.

Methods	Training Time (secs)	Testing Time (secs)	Total Time (secs)
DASS	976.54	20.41	996.95
ALSP	700.86	10.96	711.82

TABLE 5. List of abbreviations.

Abbreviation	Description
CAGR	Compounded Annual Growth Rate
DASS	Diversified Adaptive Stock Selection
DTW	Dynamic Time Warping
EWC	Elastic Weight Consolidation
GAT	Graph Attention Network
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
HConv	Hypergraph Convolution
HG	Hypergraph
LSTM	Long Short-Term Memory
SG	Simple Graph
TRM	Transformer

7) RUNNING TIME COMPARISON

Table 4 shows the average training and testing time per phase for DASS and the second-best method (ALSP). We used a computer with an Intel (R) Core (TM) i7-7700 CPU at 3.60GHz, 16GB of RAM, and a GeForce GTX 1070 Ti GPU. As shown in Table 4, the training and testing time of ALSP were 1.39 and 1.86 times faster than those of DASS. This was because DASS trained and predicted both simple graph- and hypergraph-based ranking models in non-continual and continual models, while ALSP trained and predicted only a simple graph-based ranking model in a non-continual model. Although the total time of DASS was slower than that of ALSP, it is less than 17 minutes on a commodity computer. Moreover, DASS outperformed ALSP in terms of CAGR by 23.0%P.

VI. CONCLUSION AND FUTURE WORK

We have proposed a graph learning-based framework called DASS that integrates non-continual and continual models for diversified adaptive stock selection to achieve a

harmonious balance between profit and risk. We also leveraged both simple graphs and hypergraphs for more diversity. In addition, we incorporated low-level temporal, relational, and high-level temporal modeling for graph learning to capture both temporal and relational dependencies. The performance of DASS was compared with the that of state-of-the-art stock selection methods. The results of the experiments using stocks included in the S&P500 index showed that DASS achieved a CAGR of 83.2%, outperforming the second-best method by 23.0%P. In particular, DASS exhibited stable profits and reduced risks by integrating the non-continual and continual models. In future work, we plan to combine various non-continual and continual models, incorporate more diverse graph data into our framework, and apply the framework to actual investing scenarios.

APPENDIX

See Table 5.

REFERENCES

[1] G. R. Babu, *Portfolio Management: Including Security Analysis*. New Delhi, India: Concept Publishing Company, 2007.

[2] P. Tasca, S. Battiston, and A. Deghi, "Portfolio diversification and systemic risk in interbank networks," *J. Econ. Dyn. Control*, vol. 82, pp. 96–124, Sep. 2017.

[3] K. Adam, A. Marcet, and J. P. Nicolini, "Stock market volatility and learning," *J. Finance*, vol. 71, no. 1, pp. 33–82, 2016.

[4] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[5] R. K. Nayak, D. Mishra, and A. K. Rath, "A Naïve SVM-KNN based stock market trend reversal analysis for Indian benchmark indices," *Appl. Soft Comput.*, vol. 35, pp. 670–680, Oct. 2015.

[6] L. Khaidem, S. Saha, and S. R. Dey, "Predicting the direction of stock market prices using random forest," 2016, *arXiv:1605.00003*.

[7] L. Lai, C. Li, and W. Long, "A new method for stock price prediction based on MRFs and SSVM," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 818–823.

[8] R. Kim, C. Ho So, M. Jeong, S. Lee, J. Kim, and J. Kang, "HATS: A hierarchical graph attention network for stock movement prediction," 2019, *arXiv:1908.07999*.

[9] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T.-S. Chua, "Temporal relational ranking for stock prediction," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, pp. 1–30, Apr. 2019.

[10] R. Sawhney, S. Agarwal, A. Wadhwa, T. Derr, and R. R. Shah, "Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 1, pp. 497–504.

[11] Y.-L. Hsu, Y.-C. Tsai, and C.-T. Li, "FinGAT: Financial graph attention networks for recommending top-K profitable stocks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 469–481, Jan. 2023.

[12] J.-S. Kim, S.-H. Kim, and K.-H. Lee, "Portfolio management framework for autonomous stock selection and allocation," *IEEE Access*, vol. 10, pp. 133815–133827, 2022.

[13] S. Feng, C. Xu, Y. Zuo, G. Chen, F. Lin, and J. XiaHou, "Relation-aware dynamic attributed graph attention network for stocks recommendation," *Pattern Recognit.*, vol. 121, Jan. 2022, Art. no. 108119.

[14] Y. He, Q. Li, F. Wu, and J. Gao, "Static-dynamic graph neural network for stock recommendation," in *Proc. 34th Int. Conf. Scientific Stat. Database Manage.*, Jul. 2022, pp. 1–4.

[15] H. Wang, T. Wang, S. Li, J. Zheng, S. Guan, and W. Chen, "Adaptive long-short pattern transformer for stock investment selection," in *Proc. IJCAI*, Jul. 2022, pp. 3970–3977.

[16] X. Ma, T. Zhao, Q. Guo, X. Li, and C. Zhang, "Fuzzy hypergraph network for recommending top-K profitable stocks," *Inf. Sci.*, vol. 613, pp. 239–255, Oct. 2022.

[17] T. T. Huynh, M. H. Nguyen, T. T. Nguyen, P. L. Nguyen, M. Weidlich, Q. V. H. Nguyen, and K. Aberer, "Efficient integration of multi-order dynamics and internal dynamics in stock movement prediction," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, Feb. 2023, pp. 850–858.

[18] G. Song, T. Zhao, S. Wang, H. Wang, and X. Li, "Stock ranking prediction using a graph aggregation network based on stock price and stock relationship information," *Inf. Sci.*, vol. 643, Sep. 2023, Art. no. 119236.

[19] H. Tian, X. Zhang, X. Zheng, and D. D. Zeng, "Learning dynamic dependencies with graph evolution recurrent unit for stock predictions," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 53, no. 11, pp. 6705–6717, Nov. 2023.

[20] G. Chakravorty, A. Awasthi, and B. Da Silva, "Deep learning for global tactical asset allocation," Qplum OPTrust, London, U.K., Tech. Rep., 2018. [Online]. Available: https://ssrn.com/abstract=3242432

[21] P. Zhang, Y. Yan, C. Li, S. Wang, X. Xie, G. Song, and S. Kim, "Continual learning on dynamic graphs via parameter isolation," 2023, *arXiv:2305.13825*.

[22] M. Perini, G. Ramponi, P. Carbone, and V. Kalavri, "Learning on streaming graphs with experience replay," in *Proc. 37th ACM/SIGAPP Symp. Appl. Comput.*, Apr. 2022, pp. 470–478.

[23] J. Wang, G. Song, Y. Wu, and L. Wang, "Streaming graph neural networks via continual learning," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 1515–1524.

[24] B. He, X. He, Y. Zhang, R. Tang, and C. Ma, "Dynamically expandable graph convolution for streaming recommendation," 2023, *arXiv:2303.11700*.

[25] M. Hu, Z. Tan, B. Liu, and G. Yin, "Futures quantitative investment with heterogeneous continual graph neural network," 2023, *arXiv:2303.16532*.

[26] B. Wang, Y. Zhang, J. Shi, P. Wang, X. Wang, L. Bai, and Y. Wang, "Knowledge expansion and consolidation for continual traffic prediction with expanding graphs," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 7, pp. 7190–7201, Jul. 2023.

[27] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[28] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018, pp. 1–12.

[29] S. Bai, F. Zhang, and P. H. S. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognit.*, vol. 110, Feb. 2021, Art. no. 107637.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[31] P. Senin, "Dynamic time warping algorithm review," Dept. Inf. Comput. Sci., Univ. Hawaii, Manoa, Honolulu, HI, USA, Tech. Rep., 2008, pp. 1–23, vol. 855, no. 40.

[32] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2009.

[33] S. Jin, C. Jing, Y. Wang, and X. Lv, "Spatiotemporal graph convolutional neural networks for metro flow prediction," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 43, pp. 403–409, Jun. 2022.

[34] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[35] X. Zhao, Y. Liu, Y. Xu, Y. Yang, X. Luo, and C. Miao, "Heterogeneous star graph attention network for product attributes prediction," *Adv. Eng. Informat.*, vol. 51, Jan. 2022, Art. no. 101447.

[36] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, 2017.

[37] R. Pascanu and Y. Bengio, "Revisiting natural gradient for deep networks," 2013, *arXiv:1301.3584*.

[38] J. Kim, H. Kang, and P. Kang, "Time-series anomaly detection with stacked transformer representations and 1D convolutional network," *Eng. Appl. Artif. Intell.*, vol. 120, Apr. 2023, Art. no. 105964.

[39] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mech. Syst. Signal Process.*, vol. 151, Apr. 2021, Art. no. 107398.

[40] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern Recognit.*, vol. 44, no. 9, pp. 2231–2240, Sep. 2011.

[41] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.
[42] C. Wu, F. Wu, and Y. Huang, "DA-transformer: Distance-aware transformer," 2020, *arXiv:2010.06925*.
[43] *Yahoo Finance*. Accessed: Sep. 11, 2023. [Online]. Available: https://finance.yahoo.com/
[44] W. F. Sharpe, "The Sharpe ratio," *J. Portfolio Manage.*, vol. 21, no. 1, pp. 49–58, Oct. 1994.
[45] S.-H. Kim, D.-Y. Park, and K.-H. Lee, "Hybrid deep reinforcement learning for pairs trading," *Appl. Sci.*, vol. 12, no. 3, p. 944, Jan. 2022.
[46] D.-Y. Park and K.-H. Lee, "Practical algorithmic trading using state representation learning and imitative reinforcement learning," *IEEE Access*, vol. 9, pp. 152310–152321, 2021.
[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

JAE-SEUNG KIM is currently pursuing the B.Sc. degree with Kwangwoon University, Seoul, Republic of Korea.

SANG-HO KIM received the B.S. degree in computer engineering from Kwangwoon University, Seoul, Republic of Korea, in 2023, where he is currently pursuing the integrated M.S. and Ph.D. degree in computer engineering.

KI-HOON LEE received the B.S., M.S., and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, in 2000, 2002, and 2009, respectively.

From 2010 to 2012, he was the Manager of the Advanced Institute of Technology, Korea Telecom (KT). From 2012 to 2013, he was a Senior Developer with SAP Labs Korea. In 2013, he joined Kwangwoon University, Seoul, Republic of Korea, where he is currently a Professor with the School of Computer and Information Engineering. He has published papers in leading international journals and conferences, including IEEE Access, *The VLDB Journal*, *SIGMOD Record*, and IEEE ICDE.