MCI-GRU: Stock Prediction Model Based on Multi-Head Cross-Attention and Improved GRU

Peng Zhu[a], Yuante Li[a], Yifan Hu[a], Sheng Xiang[b], Qinyuan Liu[a,*], Dawei Cheng[a], Yuqi Liang[c]

[a] *Department of Computer Science and Technology, Tongji University, Shanghai, China*
[b] *Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, Australia*
[c] *Seek Data Group, Emoney Inc., Shanghai, China*

Abstract

As financial markets become increasingly complex and the era of big data unfolds, accurate stock prediction has become more critical. Although traditional time series models, such as GRU, have been widely applied to stock prediction, they still exhibit limitations in addressing the intricate nonlinear dynamics of markets, particularly in the flexible selection and effective utilization of key historical information. In recent years, emerging methods like Graph Neural Networks and Reinforcement Learning have shown significant potential in stock prediction. However, these methods often demand high data quality and quantity, and they tend to exhibit instability when dealing with data sparsity and noise. Moreover, the training and inference processes for these models are typically complex and computationally expensive, limiting their broad deployment in practical applications. Existing approaches also generally struggle to capture unobservable latent market states effectively, such as market sentiment and expectations, microstructural factors, and participant behavior patterns, leading to an inadequate understanding of market dynamics and subsequently impact prediction accuracy. To address these challenges, this paper proposes a stock prediction model, MCI-GRU, based on a multi-head cross-attention mechanism and an improved GRU. First, we enhance the GRU model by replacing the reset gate with an attention mechanism, thereby increasing the model's flexibility in selecting and utilizing historical information. Second, we design a multi-head cross-attention mechanism for learning unobservable latent market state representations, which are further enriched through interactions with both temporal features and cross-sectional features. Finally, extensive experiments conducted on the CSI 300 and CSI 500 datasets from the Chinese stock market, as well as the NASDAQ 100 and S&P 500 datasets from the U.S. stock market, demonstrate that the proposed method outperforms the current state-of-the-art methods across multiple metrics. Furthermore, this approach has been successfully applied in the real-world operations of a fund management company, validating its effectiveness and practicality in actual financial environments.

Keywords: stock prediction, multi-head cross-attention, improved GRU, temporal features, cross-sectional features

1. Introduction

In recent years, with the advent of the big data era and the rapid development of the global economy, the complexity of financial markets [1] has significantly increased. This trend has posed unprecedented challenges to the volatility and unpredictability of stock markets [2]. Consequently, accurate stock prediction [3] has become critically important not only for investors and financial institutions, enabling them to formulate more robust investment strategies and risk management measures, but also for policymakers, who rely on these predictions for macroeconomic regulation and market oversight. Additionally, for academic researchers, stock prediction has emerged as a pivotal domain for uncovering market dynamics and behavioral patterns, thereby advancing the study of financial market theories and data-driven methodologies. These investigations not only extend the theoretical boundaries of financial economics but also provide new research directions and application scenarios for interdisciplinary fields such as machine learning

*Corresponding author

and data science [4]. Therefore, the accuracy and efficacy of stock prediction have become focal points across multiple disciplines, further stimulating extensive exploration into innovative models and methods.

Time series models [5], such as GRU and LSTM [6], have been widely utilized in stock prediction due to their significant advantages in capturing temporal dependencies within sequential data. However, these models exhibit limitations when addressing long-term dependencies in financial markets. Long-term trends and large-scale fluctuations are often obscured by noise, making it challenging for these models to extract valuable long-term dependency information from such noisy data effectively. Additionally, financial markets are characterized by a high degree of nonlinearity, with rapid shifts in market behavior driven by changes in investor sentiment, unexpected events, and other factors. These models often lack sufficient sensitivity and flexibility in handling such nonlinearity and abrupt events. Furthermore, they face challenges in the flexible selection and effective utilization of critical historical information. Given the vast and disorganized nature of data in financial markets, identifying the most relevant features for prediction has become a critical issue.

In recent years, the Transformer model [7] has demonstrated significant potential in capturing long-range dependencies and handling complex nonlinear features, owing to its architecture based on self-attention mechanisms [8]. Unlike traditional RNN models such as GRU and LSTM, the Transformer can simultaneously attend to all time steps within a sequence, making it particularly effective in extracting dependencies over extended time spans. Furthermore, the Transformer's strong parallel processing capabilities enable it to efficiently manage large-scale data, which is especially crucial when dealing with vast and diverse stock data in financial markets. However, the application of Transformer models also presents several challenges. First, the large number of parameters in Transformer models leads to high computational costs, particularly when processing ultra-large-scale financial data, potentially limiting their application in resource-constrained environments. Additionally, while the Transformer is adept at capturing complex nonlinear relationships, its performance may be compromised when confronted with highly noisy financial data. Therefore, to fully harness the advantages of the Transformer, it is often necessary to incorporate more sophisticated preprocessing and feature selection methods to enhance its accuracy and efficiency in stock prediction tasks.

Artificial intelligence technologies' rapid advancement [9], particularly in Graph Neural Networks (GNNs) [10] and Reinforcement Learning (RL) [11], has introduced unprecedented potential for stock prediction. These technologies, through innovative algorithmic designs and deep learning models, promise to enhance the ability to capture the complex dynamics of financial markets. For example, methods that employ GNNs are able to precisely capture the complex and diverse interdependencies within financial data by modeling the relationships between stocks as graph structures. This approach not only uncovers deep connections that are difficult for traditional models to reveal but also more effectively reflects the nonlinear characteristics of the market. Meanwhile, methods based on RL progressively learn and optimize trading strategies by simulating continuous interaction with the market environment. This method is highly adaptive, enabling dynamic strategy adjustments to cope with the market's rapidly changing conditions. However, despite the considerable promise of these emerging methodologies, they still face significant challenges in practical application. Firstly, these models often rely on large-scale, high-quality datasets, which are difficult to acquire or construct in real-world scenarios. When confronted with data sparsity and noise—common issues in financial markets—the predictive performance of these models can be severely compromised. Additionally, the computational complexity of these methods is high; the training process is not only time-consuming and resource-intensive but also demands substantial computational power, significantly limiting their widespread adoption in practical financial applications. More critically, a fundamental limitation of current models lies in their inability to capture unobservable latent market states effectively. Factors such as market sentiment, investor expectations, microstructural elements, and participant behavior patterns play crucial roles in shaping market dynamics. However, the failure to adequately account for these latent factors often leads to a superficial understanding of the market, thereby constraining the predictive accuracy and practical utility of these models.

To address these challenges, this paper introduces a novel stock prediction model, MCI-GRU, which integrates a multi-head cross-attention mechanism and improved GRU architecture. First, by replacing the reset gate in the traditional GRU model with an attention mechanism, the MCI-GRU model significantly improves the flexibility in selecting and utilizing historical time series information. Second, MCI-GRU employs a Graph Attention Network (GAT) to extract cross-sectional features from stock data. Additionally, this paper introduces a multi-head cross-attention mechanism designed to capture latent, unobservable market states. The model's expressive capability and ability to capture complex market dynamics are further enhanced by interacting these latent states with temporal and cross-sectional features. We conduct extensive experimental evaluations on several stock market datasets, including the CSI 300 and CSI 500 indices in China, as well as the NASDAQ 100 and S&P 500 indices in the United States. The experimental results demonstrate that the proposed MCI-GRU model outperforms existing state-of-the-art methods across multiple key performance metrics. Furthermore, the model has been successfully deployed in the operations of a leading fund management company, showcasing its practical applicability and effectiveness in real-world financial environments. In summary, the key contributions of this paper are as follows:

- This paper improves the traditional GRU model by replacing the reset gate with an attention mechanism, enabling the model to more flexibly select critical historical information, thereby enhancing the effectiveness of filtering and utilizing past sequence data.
- We propose a multi-head cross-attention mechanism to learn representations of unobservable market latent states. These learned representations are then interactively integrated with both temporal and cross-sectional features, thereby enriching the model's feature representation capacity.
- We conduct empirical studies on stock market datasets from multiple countries, and the results demonstrate that the proposed method outperforms existing state-of-the-art approaches. Moreover, the method has been successfully deployed in practical applications on a fund company's platform.

2. Related Work

Stock market prediction has been a long-standing challenge in the field of finance, commonly solutions include traditional and machine learning, deep and reinforcement learning, graph neural networks, and the latest methods.

2.1. Traditional Learning and Machine Learning Methods

Traditional approaches, including Autoregressive (AR) [12], ARIMA [13, 14], and Exponential Smoothing [15, 16], have been extensively employed in stock prediction, primarily for modeling linear trends. With advancements in computational technology, machine learning methods such as Hidden Markov Models(HMM) [17, 18], Support Vector Machines (SVM) [19, 20], K-Nearest Neighbor (KNN) [21, 22], Decision Trees [23], and Neural Networks [24, 25, 26], including Single Layer Perceptron (SLP) and MultiLayer Perceptron (MLP) [27, 28], have garnered significant attention for their ability to model complex patterns in stock data. For instance, [29] introduced a decision tree-based approach demonstrating the efficacy of Random Forests in short-term prediction and the superior long-term accuracy of J48 combined with Bagging. Similarly, [30] utilized high-order Hidden Markov Models with advanced parameters, including state transition probabilities dependent on multiple previous states and observation probabilities modeled as Gaussian mixtures. This method incorporated a state dimension reduction technique to simplify parameter estimation and decoding, alongside a dynamic trading strategy based on identified hidden states, validated on the CSI 300 and S&P 500 indices. Moreover, [31] integrated feature-weighted support vector machine (FWSVM) and feature-weighted k-nearest neighbor (FWKNN) techniques by calculating feature importance via information gain, which informed weight assignment in SVM classification and distance calculation in KNN.

While these advanced machine learning techniques have demonstrated their capacity to capture complex nonlinear interactions within stock data, challenges remain, including susceptibility to overfitting due to the low signal-to-noise ratio, high trading volumes, frequent trading, significant price volatility, and the multitude of influencing factors inherent in financial markets.

2.2. Deep Learning and Reinforcement Learning Methods

With the rapid advancement of deep learning, this technology has been extensively applied to stock prediction in financial markets, producing notable outcomes [32, 25]. Recurrent Neural Networks (RNNs) [33, 34, 35] have demonstrated exceptional capabilities in this field by effectively modeling long-term dependencies in time series data, utilizing inputs such as stock prices to forecast market trends. Recent studies have introduced sophisticated models aimed at further enhancing prediction accuracy. For instance, [36] proposed the StockNet model based on GRU, which incorporates an injection module to mitigate overfitting and a survey module for comprehensive stock analysis. Furthermore, [37] integrated convolutional LSTM units with a sequence-to-sequence framework and attention mechanisms, employing variational methods and backward decoders to improve prediction accuracy and robustness.

Similarly, [38] advanced the attention-based LSTM model by implementing adversarial training to enhance its generalization capability.

Deep learning models often demonstrate instability when confronted with extreme market fluctuations, such as those experienced during the 2008 financial crisis [39] and the 2019 COVID-19 pandemic [40]. In response, reinforcement learning models have gained prominence due to their adaptability and capacity for continuous learning. Reinforcement learning approaches in investment strategies can be broadly categorized into two types: value-based and policy-based [41, 42]. Value-based approaches involve learning a critique to estimate the expected outcomes of trading actions within the market. Common value-based approaches in investment strategies include Q-learning [43], Deep Q-learning [44, 45], Recurrent Reinforcement Learning [46, 47], and Sarsa [48, 49]. However, a major limitation of value-based approaches is the complexity of accurately approximating the market environment with a critic. As a result, policy-based approaches [50, 51] are often regarded as more suitable for financial markets. For example, [52] integrated deep attention networks with reinforcement learning, optimizing parameters through discrete agent actions to maximize the Sharpe ratio of investments. To further balance profit and risk, [53] introduced a multi-objective deep reinforcement learning (MODRL) approach for intraday trading of stock index futures, combining deep learning for feature extraction with reinforcement learning for decision-making. Despite their potential, reinforcement learning models face challenges, such as the requirement for large datasets and difficulties in model interpretability, which can hinder their practical application in financial markets.

2.3. Graph Neural Networks and Latest Methods

In recent years, Graph Neural Networks [54, 55, 56] have garnered significant attention in stock prediction due to their capacity to capture complex interdependencies within financial data. For instance, [57] proposed a hybrid model that integrates Recurrent Neural Networks with GNN, facilitating real-time predictions. Similarly, [58] introduced a hierarchical attention mechanism into GNN, thereby improving the model's ability to analyze multi-level market dependencies and perform structured analyses of stock trends. To capture a stock's intrinsic value more accurately, [59] developed a Higher-order Graph Attention Network (H-GAT), which differentiates itself by modeling complex subgraph structures involving more than two stocks and incorporating both technical and fundamental factors. This approach contrasts with traditional GNNs, which typically consider only simple pairwise relationships, thus enhancing the model's ability to reflect the intrinsic value of stocks. However, many graph-based models frequently overlook the diversity of stock price changes and the temporal dynamics inherent in these fluctuations, necessitating the development of more innovative graph-based approaches. For instance, [60] introduced a market-guided stock transformer that can dynamically simulate the instantaneous and cross-temporal correlations among stocks, leading to enhanced accuracy in stock trend predictions. Additionally, [61] successfully integrated long-term trends, short-term fluctuations, and sudden events into a cohesive graph-based framework, significantly surpassing traditional methods by accounting for the multi-scale nature of market dynamics. Despite their advantages in stock prediction, Graph Neural Networks exhibit several limitations. These include insufficient modeling capabilities for complex nonlinear relationships and anomalous scenarios within the stock market, as well as a lack of robustness to data sparsity and noise. Furthermore, certain graph-based models may be prone to overfitting, particularly in contexts characterized by limited or imbalanced training data.

With the rapid advancements in Large Language Models (LLMs) [62, 63, 64], their application in stock prediction has garnered significant scholarly interest. LLMs, such as GPT-4, have demonstrated remarkable capabilities in natural language understanding, making them highly suitable for financial sentiment analysis and predictive modeling. Research by [65] revealed a strong correlation between sentiment analysis generated by ChatGPT for news headlines and subsequent daily stock market returns, highlighting the potential of LLMs in capturing market sentiment and its impact on stock prices. This study underscores the utility of LLMs in extracting and quantifying sentiment from unstructured textual data, which can be critical for short-term stock forecasting. Moreover, the integration of LLMs with Graph Neural Networks has opened new avenues for enhancing stock prediction accuracy. For example, [66] employed ChatGPT to infer dynamic network structures from financial news, which were subsequently incorporated into a GNN for stock prediction. This approach not only leverages the linguistic prowess of LLMs in understanding and summarizing complex financial information but also capitalizes on the GNN's ability to model intricate relationships between stocks. The resulting hybrid model demonstrated superior predictive performance, suggesting that the synergistic combination of LLMs and GNNs can effectively address the challenges of dynamic and interconnected financial markets.

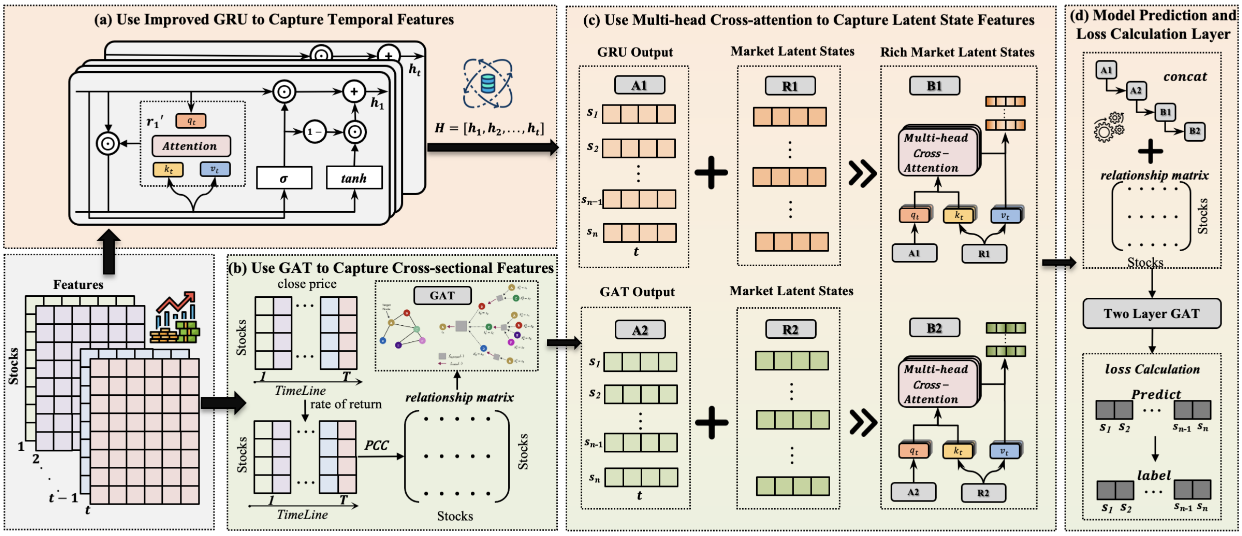


Figure 1: The architecture of the proposed model MCI-IGRU. The enhanced GRU model in Part (a) integrates an attention mechanism in place of the reset gate, greatly improving its ability to capture and learn from temporal patterns. Part (b) leverages attention mechanisms to identify and weigh relationships between stocks, effectively extracting cross-sectional features. Part (c) captures latent market conditions influencing stock behavior, allowing the model to learn and represent hidden, non-observable market states. In part (d), the final prediction process integrates these learned features, refining outcomes through optimized loss calculation for improved performance.

3. Methods

In this section, we give a detailed introduction to the MCI-GRU model proposed in this paper and the model structure is shown in Figure 1. The whole model is divided into four modules: Use Improved GRU to Capture Temporal Features (Part a), Use GAT to Capture Cross-sectional Features (Part b), Use Multi-head Cross-attention to Capture Latent State Features (Part c), Model Prediction and Loss Calculation Layer (Part d). Part(a) employs an enhanced GRU model that replaces the reset gate with an attention mechanism, significantly enhancing the model's capacity to represent and learn from temporal data. Part(b) enhances the model by using attention mechanisms to capture and weigh the relationships between different stocks, thereby extracting cross-sectional features from the data. Part(c) captures hidden market conditions that affect stock behavior, enabling the model to learn and characterize latent, non-directly observable market state features. Part(d) refines the final predictions by integrating the learned features and calculating the loss to optimize model performance. In the following subsections, we first give the pre-definition and then introduce each module in detail.

3.1. Predefinition

We consider a collection of stocks denoted by the set $S = \{s_1, s_2, \cdots, s_N\}$, where each s_i represents an individual stock, and N is the total number of stocks within the dataset. For any stock s_i, the data associated with the t-th trading day is represented by the vector $x_{it} = \left\{x_{it}^{open}, x_{it}^{close}, x_{it}^{high}, x_{it}^{low}, x_{it}^{volume}, x_{it}^{turnover}\right\}$, where x_{it}^{open}, x_{it}^{close}, x_{it}^{high}, x_{it}^{low}, x_{it}^{volume} and $x_{it}^{turnover}$ correspond to the opening price, closing price, highest price, lowest price, trading volume, and turnover amount on day t, respectively. We let d_x denote the number of features used to describe each stock on each day. The time series data for stock s_i over t days is encapsulated in the set $x_i = \{x_{i1}, x_{i2}, \ldots, x_{it}\}$. Collectively, the dataset for all stocks is represented as $X = \{x_1, x_2, \ldots, x_N\}$.

3.2. Use Improved GRU to Capture Temporal Features

In time series prediction tasks, the GRU model has been widely employed due to its capability to capture temporal dependencies within sequential data effectively. However, traditional GRU models have certain limitations in capturing complex temporal relationships, particularly when dealing with long-term dependencies, where they may struggle to extract deeper features from the sequence. To address this, the present work utilizes an enhanced GRU model, incorporating an attention mechanism in place of the reset gate, thereby improving the model's ability to represent and learn from temporal data.

3.2.1. Basic Structure of the GRU

In the classical GRU model, the hidden state h_t is updated through two gating mechanisms: the update gate z_t and the reset gate r_t. The formulas are as follows:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \end{aligned} \tag{1}$$

where $x_t \in \mathbb{R}^{d_x}$ denotes the input of the stock at the current time step, $h_{t-1} \in \mathbb{R}^{d_h}$ represents the hidden state from the previous time step and d_h represents the dimension of the hidden state. $W_z \in \mathbb{R}^{d_h \times d_x}$, $U_z \in \mathbb{R}^{d_h \times d_h}$, $W_r \in \mathbb{R}^{d_h \times d_x}$ and $U_r \in \mathbb{R}^{d_h \times d_h}$ are weight matrices. $b_z \in \mathbb{R}^{d_h}$ and $b_r \in \mathbb{R}^{d_h}$ are bias terms, and σ is the activation function, typically the sigmoid function. $z_t \in \mathbb{R}^{d_h}$ and the reset gate $r_t \in \mathbb{R}^{d_h}$ is employed to regulate the influence of the previous time step's hidden state h_{t-1} when computing the candidate's hidden state $\tilde{h}_t \in \mathbb{R}^{d_h}$ at the current time step, as detailed by the following formula:

$$\tilde{h}_t = \tanh(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h) \tag{2}$$

where $W_h \in \mathbb{R}^{d_h \times d_x}$, $U_h \in \mathbb{R}^{d_h \times d_h}$ are weight matrices. $b_h \in \mathbb{R}^{d_h}$ is bias terms and $\odot$ denotes element-wise multiplication. Ultimately, the hidden state h_t at the current time step is controlled by the update gate z_t, as shown below:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{3}$$

In these equations, the reset gate r_t determines how much the previous time step's hidden state h_{t-1} is "reset" at the current time step. However, this mechanism exhibits certain limitations in capturing long-term dependencies.

3.2.2. Introduction of the Improved GRU

To overcome the aforementioned limitations, the proposed model introduces an attention mechanism to replace the traditional reset gate r_t in GRU. The attention mechanism dynamically allocates weights to different time steps in the sequence, thereby capturing critical information within the time series data more precisely. Specifically, the traditional reset gate r_t is replaced with an attention-based weight coefficient α_t, computed as follows:

$$\alpha_t = Attention(h_{t-1}, x_t) \tag{4}$$

where $\alpha_t \in \mathbb{R}$ is the attention weight vector. The core idea of the attention mechanism is to allocate attention weights by calculating the similarity between the query, key, and value. In this model, the hidden state $h_{t-1} \in \mathbb{R}^{d_h}$ from the previous time step is treated as the query, while the current time step's input x_t serves as both the key and value. The attention calculation involves the following steps:

- Linear Transformations of Query, Key, and Value: First, the hidden state h_{t-1} and the input x_t are linearly transformed into query, key, and value spaces respectively:

$$q_t = W_q h_{t-1}, \quad k_t = W_k x_t, \quad v_t = W_v x_t \tag{5}$$

  where $W_q \in \mathbb{R}^{d_q \times d_h}$, $W_k \in \mathbb{R}^{d_k \times d_h}$ and $W_v \in \mathbb{R}^{d_v \times d_h}$ are learnable linear transformation matrices. $q_t \in \mathbb{R}^{d_q}$ and d_q represent the dimensions of the query, which is used to calculate the attention weight. $k_t \in \mathbb{R}^{d_k}$ and d_k represents the dimensions of key. $v_t \in \mathbb{R}^{d_v}$ and d_v represents the dimension of value, which is usually the same as d_q and d_k, that is, $d_q = d_k = d_v$.

- Attention Weight Calculation: The attention weights are obtained by computing the dot product similarity between the query and key:

$$\alpha_t = softmax(\frac{q_t k_t^\top}{\sqrt{d_k}}) \tag{6}$$

  where d_k is used to scale the dot product results to prevent numerical instability.

- Weighted Sum: The final reset gate value is derived from a weighted sum of the values, using the attention weights:

$$r_t^{'} = \alpha_t v_t \tag{7}$$

where the new $r_t^{'} \in \mathbb{R}^{d_v}$ dynamically selects the most important parts of the current input x_t and the previous hidden state h_{t-1}, thereby enhancing the model's ability to capture long-term dependencies.

3.2.3. Update of the Hidden State

With the new reset gate $r_t^{'}$, the update formula for the GRU's hidden state is adjusted as follows:

$$\begin{aligned} \tilde{h}_t^{'} &= \tanh(W_h x_t + r_t^{'} \odot (U_t h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t^{'} \end{aligned} \tag{8}$$

where z_t is the update gate. This updated hidden state calculation integrates the dynamic information selection capability provided by the attention mechanism, allowing the model to better extract long-term dependency information and key features from the time series data.

3.2.4. Final Output

Through recursive computation over multiple time steps, the enhanced GRU model generates a final sequence of hidden states $H = [h_1, h_2, \ldots, h_t]$, where each h_t incorporates information from past time steps, with increased focus on important time steps due to the attention mechanism. For subsequent processing, the final hidden state h_t is typically taken as the representation vector for the entire sequence, denoted as $A_1 \in \mathbb{R}^{N \times d_h}$. This output A_1 is then used as input for further feature extraction and model learning in the next processing stage.

3.3. Use GAT to Capture Cross-sectional Features

The Graph Attention Network is a key component in the model architecture, responsible for extracting cross-sectional features from the data by capturing the relationships between different stocks. GAT extends the traditional Graph Convolutional Network by incorporating attention mechanisms, allowing the model to assign different levels of importance to different nodes (stocks) in the graph based on their relationships.

3.3.1. Input Representation

In this model, the input to the GAT layer is a matrix representing the features of all stocks at a specific time step. The input matrix has dimensions (N, d_x), N is the number of stocks that represent the nodes in the graph, and d_x is the dimensionality of the feature vector for each stock. This input representation is derived from the origin stock data. Hence, the GAT layer particularly focuses on capturing the cross-sectional dependencies between stocks.

3.3.2. Graph Construction

The graph in the GAT layer is constructed where each node represents a stock, and the edges between nodes represent the relationships between these stocks. The edges' weights, or the relationships' strengths, are determined by the historical correlations of the stocks' returns over the past year. These correlations are typically computed using Pearson correlation or other statistical measures. We calculate the historical correlation of stock returns over the past year for any two stocks s_i and s_j to determine the strength of their relationship. First, let $r_i(t')$ and $r_j(t')$ represent the returns of stocks s_i and s_j at time t, where $t' = 1, 2, \ldots, T$ denotes the number of trading days in the past year (e.g., 252 trading days per year). The returns are typically calculated using the log return formula:

$$r_i(t') = \frac{x_{it'}^{close} - x_{i(t'-1)}^{close}}{x_{i(t'-1)}^{close}} \tag{9}$$

where $r_i(t')$ represents the closing price of the stock s_i at time t'. Next, we compute the Pearson correlation coefficient $\rho(s_i, s_j)$ between the return series of the two stocks, which measures their linear correlation. The Pearson correlation coefficient is given by the following formula:

$$\rho(s_i, s_j) = \frac{\sum_{t'=1}^{T}(r_i(t') - \bar{r}_i)(r_j(t') - \bar{r}_j)}{\sqrt{\sum_{t'=1}^{T}(r_i(t') - \bar{r}_i)^2}\sqrt{\sum_{t'=1}^{T}(r_j(t') - \bar{r}_j)^2}} \tag{10}$$

where $\bar{r}_i$ and $\bar{r}_j$ represent the average returns of stocks s_i and s_j over the past year. Based on this correlation coefficient $\rho(s_i, s_j)$, we assign an edge weight to represent the strength of the relationship between stocks s_i and s_j. The weight $w_{i,j}$ is typically set to $\rho(s_i, s_j)$. To optimize the learning process, not all relationships are included in the graph. Instead, threshold-based filtering is applied using a parameter known as $judge_{value}$. This parameter allows the model to retain only edges that represent significant relationships, effectively reducing noise and focusing the model on the most relevant connections.

3.3.3. Attention Mechanism in GAT

The core of the GAT layer lies in its attention mechanism, which dynamically computes the importance (attention coefficients) of each node's neighbors when aggregating information. For each node i in the graph, the GAT layer performs the following operations:

- Linear Transformation: Each node's feature vector x_{it} is linearly transformed using a learnable weight matrix W_g:

  $$h_i^{'} = W_g x_{it} \tag{11}$$

  where x_{it} is the date of the stock s_i in time t, $W_g \in \mathbb{R}^{d_g \times d_x}$, $h_i^{'} \in \mathbb{R}^{d_g}$ is the transformed feature vector. d_g is the dimension of the hidden layer.

- Attention Coefficients Calculation: The attention coefficients between stock i and its neighbor j are computed using the following equation:

  $$e_{ij} = LeakyReLU(a^{\top}[h_i^{'} \| h_j^{'}]) \tag{12}$$

  where $a \in \mathbb{R}^{2d_g}$ is a learnable attention vector, $\|$ denotes concatenation, *LeakyReLU* is a non-linear activation function that introduces non-linearity to the attention computation.

- Normalization: The attention coefficients are then normalized across all neighbors of node i using the softmax function:

  $$\sigma_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in N(i)} exp(e_{ik})} \tag{13}$$

  where σ_{ij} is the normalized attention score between stock i and stock j, N_i denotes the set of neighbors of the node i.

- Feature Aggregation: Finally, the node's updated representation is computed as a weighted sum of its neighbors' transformed features, weighted by the normalized attention coefficients:

  $$h_i^{''} = \sigma(\sum_{j \in N(i)} \sigma_{ij} h_j^{'}) \tag{14}$$

  where σ is a non-linear activation function, typically ReLU.

3.3.4. Output of the GAT Layer

The final output of the GAT layer is a matrix $A_2 \in \mathbb{R}^{N \times d_g}$, where each row corresponds to the updated feature vector of stock, now enriched with information from its neighbors. This output captures the cross-sectional dependencies between stocks and serves as the input to the next stage of the model, where the market latent state learning layer further processes these features.

3.4. *Use Multi-head Cross-attention to Capture Latent State Features*

The Market Latent State Learning Layer is a crucial component of the model, designed to capture and represent the latent states of the market that are not directly observable from the raw data. This layer is specifically tailored to model the underlying market conditions that influence stock behavior, allowing the model to better understand and predict stock movements by leveraging these hidden states.

3.4.1. *Initialization of Market Latent States*

The process begins with the initialization of a set of learnable market latent state vectors. These vectors are meant to represent different latent market conditions or factors that could be driving stock prices. The initialization is as follows:

- Number of Latent States (d_r): The model initializes d_r latent state vectors. The number d_r is a hyperparameter that can be tuned based on the complexity of the market being modeled. A larger d_r allows the model to capture a greater variety of latent factors.

- Dimensionality (d_i): Each latent state vector is of dimensionality d_i, which should match the dimensionality of the stock features that the model has learned so far. This ensures compatibility when these latent states interact with the outputs from previous layers.

The initialized latent state vectors are denoted as R_1 and R_2, corresponding to the two main types of information the model processes: time series information (from the Improved GRU output A_1) and cross-sectional information (from the GAT output A_2). The dimensions of R_1 and R_2 are both (d_r, d_i).

3.4.2. *Multi-head Cross-Attention Mechanism*

The core of the market latent state learning process involves a multi-head cross-attention mechanism. This mechanism allows the latent state vectors to interact with the outputs from the Improved GRU and GAT layers, effectively absorbing the relevant information from these outputs and refining the latent state representations.

Multi-head Cross-Attention between R_1 and A_1: In the multi-head cross-attention mechanism, the interaction between A_1 and R_1 can be modeled through the cross-attention process, where A_1 is used as the query and R_1 is used as the key and value. Here's how the multi-head cross-attention mechanism operates between A_1 and R_1:

- Linear Transformations: For each attention head i, we compute the attention scores between A_1(Query) and R_1(Key and Value) as follows:

$$Q_i = A_1 W_i^Q, \quad K_i = R_1 W_i^K, \quad V_i = R_1 W_i^V \tag{15}$$

  where $A_1 \in \mathbb{R}^{N \times d_h}$, $R_1 \in \mathbb{R}^{d_r \times d_i}$ and we set $d_h = d_i$. For head i, $W_i^Q \in \mathbb{R}^{d_h \times d_{h'}}$ is the learnable weight matrix for queries, $W_i^K \in \mathbb{R}^{d_h \times d_{h'}}$ is the learnable weight matrix for keys, $W_i^V \in \mathbb{R}^{d_h \times d_{h'}}$ is the learnable weight matrix for values. $Q_i \in \mathbb{R}^{N \times d_{h'}}$, $K_i \in \mathbb{R}^{d_r \times d_{h'}}$ and $V_i \in \mathbb{R}^{d_r \times d_{h'}}$. $d_{h'} = \frac{d_h}{h'}$ is the dimension of each attention head and h' is the number of attention heads.

- Scaled Dot-Product Attention: For each head i, we compute attention scores between the query A_1 and the key R_1 as follows:

$$head_i = Attention(Q_i, K_i, V_i) = softmax(\frac{Q_i K_i^\top}{\sqrt{d_{h'}}})V_i \tag{16}$$

  where $Q_i K_i^\top \in \mathbb{R}^{N \times d_r}$ gives the attention weights, $\frac{1}{\sqrt{d_{h'}}}$ is the scaling factor to avoid overly large dot-product values. The softmax function ensures that the attention weights sum to 1.

- Concatenation of Attention Heads: After computing attention for each of the h' heads, the outputs of all heads are concatenated:

$$B_1 = MultiHead(A_1, R_1) = Concat(head_1, \ldots, head_{h'})W^O \tag{17}$$

where W^O is a learnable matrix used to project the concatenated results into the output space. The final output after applying the multi-head cross-attention mechanism between A_1 and R_1 is $B_1 \in \mathbb{R}^{N \times d_h}$, which captures complex relationships between the two sets of features.

Multi-head Cross-Attention between R_2 and A_2: The second multi-head cross-attention operation is performed similarly, but this time between the latent state vectors R_2 and the GAT output A_2. By a similar method, we calculated $B_2 \in \mathbb{R}^{N \times d_g}$.

3.4.3. Integration of Market Latent States

After the multi-head cross-attention mechanisms have been applied, the outputs B_1 and B_2 are considered to be enriched latent state representations. These vectors now capture the essential time series and cross-sectional features of the market, making them powerful representations for subsequent tasks, such as predicting stock movements or identifying market regimes.

3.5. Model Prediction and Loss Calculation Layer

The Loss Calculation Layer is the final stage of the model, responsible for synthesizing the outputs from previous layers and generating predictions. This layer also defines how the model is trained by computing the difference between the predicted values and the ground truth, which is then minimized during the training process.

3.5.1. Input Composition

The inputs to the Loss Calculation Layer are the outputs from three key components of the model: the time-series representation A_1, the cross-sectional representation A_2, and the latent market state representations B_1 and B_2. Time-Series Representation A_1 comes from the improved GRU layer, which captures time-series dependencies in the stock data. Cross-sectional representation A_2 comes from the GAT layer, which models the relationships between stocks based on their inter-stock correlations. Latent Market State Representations B_1 and B_2 come from the Market Latent State Learning Layer, representing the hidden market factors learned from the time series and cross-sectional data.

To prepare for the final prediction, the model concatenates these outputs to form a comprehensive feature vector that integrates all relevant information. The concatenation can be expressed as:

$$Z = Concat(A_1, A_2, B_1, B_2) \tag{18}$$

where $Z \in \mathbb{R}^{N \times d_z}$ is the combined feature vector that will be used for the final prediction, and $d_z = 2d_h + 2d_g$ depends on the individual dimensions of A_1, A_2, B_1 and B_2.

3.5.2. Final Prediction with GAT Layers

Once the feature vector Z is obtained, it is passed through additional GAT layers to make the final prediction. The purpose of these layers is to refine the concatenated features by considering the relationships between stocks (nodes) in a graph structure, as GAT layers are well-suited to modeling graph-structured data. The GAT layers in this part of the model function similarly to the earlier GAT layers, but they now operate on a more comprehensive set of features, combining both temporal and cross-sectional information, as well as the latent market state representations. The structure of the GAT layers can be described as follows:

- Graph Construction: The graph's structure is the same as in the earlier GAT layer. The nodes represent individual stocks, and the edges represent relationships between the stocks based on their historical correlations over the past year. These correlations are filtered using the judge_value threshold, a tunable parameter that determines which relationships are included in the graph.

- Attention Mechanism: The attention mechanism calculates the importance of each stock's neighbors using the following equation:

$$e'_{ij} = LeakyReLU(a'^{\top})[W_z Z_i \| W_z Z_j]) \tag{19}$$

where $Z_i \in \mathbb{R}^{d_z}$ and $Z_j \in \mathbb{R}^{d_z}$ are the concatenated feature vectors of stock i and stock j, $W_z \in \mathbb{R}^{d_{z'} \times d_z}$ is a learnable weight matrix, a' is a learnable attention vector, $\|$ represents concatenation, $d_{z'}$ is the size of the dimension.. The attention scores are then normalized across all neighboring stocks using a softmax function:

$$\sigma'_{ij} = \frac{\exp(e'_{ij})}{\sum_{k \in N'(i)} exp(e'_{ik})} \tag{20}$$

where N'_i denotes the set of neighbors of the node i.

- Feature Aggregation: The final output for each stock is computed as a weighted sum of its neighbors' features, with the attention weights σ'_{ij} determining the contribution of each neighbor:

$$Z'_i = \sum_{j \in N'(i)} \sigma'_{ij} W_z Z_j \tag{21}$$

where $Z'_i \in \mathbb{R}^{d_{z'}}$ effectively integrates information from neighboring stocks, leading to a refined feature representation that incorporates both temporal, cross-sectional, and latent market features.

- Dimensionality Reduction: Put the output Z'_i of the first GAT layer into the second GAT layer to reduce the dimension. The calculation method is the same as above, and the final output Z''_i is obtained as the prediction result of each stock.

3.5.3. Loss Function

After the final prediction is obtained from the GAT layers, the next step is to compute the loss, which measures the difference between the model's predicted stock returns and the actual values. The choice of the loss function is critical, as it guides the model's training process and influences its performance. In this model, we use mean squared error (MSE) for stock prediction tasks. It is defined as:

$$Loss = \frac{1}{N} \sum_{i=1}^{N} (Z''_i - y_i)^2 \tag{22}$$

where Z''_i is the predicted value for the stock i and y_i is the actual value. During training, the model parameters are optimized to minimize the chosen loss function, leading to better predictive performance over time. We use Adam as a gradient-based optimization algorithm to update the model's parameters.

4. Experiments

In this section, we will thoroughly discuss the experimental design, including the experimental setup, baseline models, results, parameter sensitivity analysis, ablation studies, and case analysis. The experimental setup section specifically covers the datasets, evaluation metrics, and model parameters.

4.1. Experimental Setttings

4.1.1. ***Datasets***

We employ four distinct stock market datasets to rigorously assess the robustness and generalizability of our model across varying market conditions, the details of the dataset are shown in Table 1. Our selection encompasses the Shanghai-Shenzhen CSI 300[1] and CSI 500[2] datasets, which provide comprehensive coverage of the Chinese market's large-cap and mid-cap sectors, respectively. These datasets enable a detailed exploration of the dynamics within one of the world's largest and most complex financial markets. In contrast, the S&P 500[3] dataset represents 500 leading

[1]https://cn.investing.com/indices/csi300
[2]https://cn.investing.com/indices/china-securities-500-historical-data
[3]https://hk.finance.yahoo.com/quote/%5EGSPC/history/

Table 1: Details of the CSI 300, CSI 500, NASDAQ 100, and S&P 500 dataset.

Stock Market	CSI300	CSI500
Number of Stocks	285	450
Year Established	2005	2007
Industry Coverage	Broad Coverage	Broad Coverage
Market Cap Range	Large Cap	Mid & Small Cap
Representative Companies	ICBC, Vanke	Citic Bank, Shuanghui
Market	China A-Share	China A-Share
Calculation Method	Free-float Market Cap Weighted	Free-float Market Cap Weighted
Coverage Ratio	Covers 70% of China's A-share market cap	Covers 85% of China's A-share market cap
Stock Market	NASDAQ100	S&P500
Number of Stocks	99	498
Year Established	1985	1957
Industry Coverage	Primarily Tech	Broad Coverage
Market Cap Range	Large Cap	Large Cap
Representative Companies	Apple, Microsoft	Apple, Amazon
Market	US Stock Market	US Stock Market
Calculation Method	Free-float Market Cap Weighted	Free-float Market Cap Weighted
Coverage Ratio	Covers 90% of NASDAQ's total market cap	Covers 80% of NYSE and NASDAQ total market cap

companies across the U.S. market, offering insights into a broad and diverse economic landscape. Additionally, the NASDAQ 100[4] dataset highlights the top 100 non-financial firms listed on the NASDAQ, with a particular focus on the technology sector's rapidly evolving dynamics. Collectively, these datasets provide a comprehensive view of different market behaviors and geographic regions, thereby supporting a robust evaluation of our model's predictive capabilities across varied financial contexts.

We structure our datasets following a temporal sequence, dividing them into distinct phases for training (from January 1, 2018, to December 31, 2021), validation (from January 1, 2022, to December 31, 2022), and testing (from January 1, 2023, to December 31, 2023). In our forecasting approach, we employe features derived from the previous 60 trading days to predict stock return rankings over the next 21 trading days. This methodology closely mirrors the decision-making process in real-world trading scenarios. For the baseline analysis, we utilize data from the four aforementioned stock markets, concentrating on six key financial indicators: open price, close price, high price, low price, turnover, and volume. We commence by implementing procedures for outlier detection and normalization to ensure data integrity and reduce the impact of anomalous values. Following this, we calculate the daily return for each stock as the label during training, defined as the percentage change between the closing prices of consecutive trading days.

4.1.2. ***Evaluation Metrics***

The trading strategy simulated by our model is outlined as follows:

- At the close of trading day t, the model generates a prediction score for each stock, ranking them based on the expected rate of return.
- At the opening of trading day $t + 1$, traders liquidate the stocks purchased on day t and acquire those ranked in the top-k for expected returns.
- If a stock consistently ranks among the highest expected returns, it remains in the trader's portfolio.
- Transaction costs are excluded from consideration in this simulation.

In order to improve the reliability of the evaluation, we conduct ten training and predictions for each method and take the average result of the ten times as the final prediction result, and then trade the strategy.

[4]https://hk.finance.yahoo.com/quote/%5EIXIC/history

The primary objective is to identify stocks with the highest returns and to evaluate the performance of both the baseline and our proposed model, we employ six key financial metrics:

- Annualized Rate of Return (ARR): This core metric aggregates the daily returns of selected stocks over a year, indicating the effectiveness of the investment strategy. It is computed as $ARR = (\prod_{T}^{t+1}(1 + r_t))^{\frac{252}{T}} - 1$, where r_t is the daily return and T is the total number of trading days in the year.
- Annualized Volatility (AVoL): This metric captures the annualized standard deviation of daily returns, representing the risk associated with the strategy. It is calculated as $AVoL = std(\frac{P_t}{P_{t-1}} - 1) * \sqrt{252}$, where P_t and P_{t-1} are the stock prices on day t and day $t-1$, respectively.
- Maximum Drawdown (MDD): MDD measures the most substantial decline from a peak to a trough during the testing period, indicating the strategy's potential risk of loss. It is computed as $\underset{t\in[1,T]}{max}(\frac{max_{i\in[1,t]}P_i - P_t}{max_{i\in[1,t]}P_i})$, where P_t is the price of the stock at time t, and T is the length of the period.
- Annualized Sharpe Ratio (ASR): This metric evaluates the return per unit of volatility and is calculated as $ASR = \frac{ARR}{AVoL}$, reflecting the risk-adjusted performance of the strategy.
- Calmar Ratio (CR): CR assesses the return relative to the maximum drawdown, calculated as $CR = \frac{ARR}{|MDD|}$, offering insight into the return-risk trade-off.
- Information Ratio (IR): This metric measures the excess return per unit of additional risk, further refining the assessment of the strategy's risk-adjusted performance. It is calculated as $IR = \frac{mean(r_t - r_{f,t})}{std(r_t - r_{f,t})}$, where r_t is the portfolio return at the time t and $r_{f,t}$ is the return of a benchmark or risk-free asset at the same time.

Together, these metrics form a comprehensive framework for evaluating both the performance and risk profile of the investment strategies. Higher values of ARR, ASR, CR, and IR, coupled with lower values of AVoL and MDD, signify superior performance.

4.1.3. ***Parameter Settings***

We configure the time window t to encompass the preceding 10 days of historical stock data as the training input for the model. The model architecture comprises four primary components, each designed to capture distinct patterns within the financial time series.

In the "Use Improved GRU to Capture Temporal Features" module, we incorporate an attention mechanism to replace the conventional reset gate in the GRU framework. This module consists of two layers, with the first layer containing 32 neurons and the second layer comprising 10 neurons. The module's output, denoted as A_1, is responsible for extracting temporal dependencies from the stock data across the historical window.

The "Use GAT to Capture Cross-sectional Features" is structured with two layers, each with tunable parameters. The initial GAT layer consists of 32 neurons and employs 4 attention heads, while the second GAT layer reduces the feature dimensionality to 4 neurons. The relationships between nodes in the graph, representing stocks, are defined based on the correlation of stock returns over the previous year. A threshold of 0.8 is applied to filter stock pairs with significant correlations during training. The output from the GAT layer, denoted as A_2, captures the structural dependencies inherent in the stock correlation graph.

In the "Use Multi-head Cross-attention to Capture Latent State Feature", two learnable market hidden state vectors, R_1 and R_2, each of dimension 32, are initialized. A multi-head cross-attention mechanism is employed, where R_1 interacts with A_1 and R_2 interacts with A_2, resulting in representation vectors B_1 and B_2, respectively. The head of the multi-head cross-attention is 4. These representation vectors encapsulate the latent market states by leveraging both temporal and structural characteristics derived from the stock data.

The "Model Prediction and Loss Calculation Layer" integrates the outputs A_1, A_2, B_1 and B_2, which are concatenated and passed through a two-layer GAT for final prediction. This final module is responsible for forecasting future stock trends by fusing both temporal and relational information.

The model is trained with a batch size of 32. The mean squared error (MSE) is adopted as the loss function, and the Adam optimizer is utilized with an initial learning rate of 0.0002. For practical application, each trading day a virtual investment portfolio is constructed, comprising the top 10 stocks ranked by the model's predicted returns.

4.2. Baseline Models

We conduct a comparative analysis between our proposed MCI-GRU model and a range of baseline models, which include prominent approaches in both traditional machine learning and deep learning for time series prediction, as well as reinforcement learning for portfolio management.

- **BLSW** [67]: Implements a mean reversion trading strategy, which assumes that asset prices will revert to their historical average over time, making it particularly effective in markets with cyclical behavior.
- **Cross-Sectional Mean Reversion (CSM)** [61]: Adopts a momentum-based approach by identifying assets that exhibit persistent price trends, and positioning trades in alignment with these trends, thereby capitalizing on short-term market movements.
- **LSTM** [68]: A widely used recurrent neural network model for time series forecasting, which captures temporal dependencies through its memory cell mechanism.
- **ALSTM** [69]: An enhanced version of LSTM, which incorporates dual attention mechanisms, one for adaptively selecting relevant features and another for focusing on significant time steps, thereby improving prediction accuracy by concentrating on key information.
- **GRU** [70]: A simplified variant of LSTM, using fewer gating mechanisms to streamline the learning process and improve computational efficiency while maintaining strong performance in sequence prediction tasks.
- **Transformer** [71, 72]: Utilizes a multi-head self-attention mechanism to capture long-range dependencies in time series data, with the ability to process entire sequences in parallel, offering scalability and improved performance over recurrent architectures.
- **TRA** [73]: Introduces a novel dynamic routing mechanism within the Transformer architecture, enabling the model to adaptively learn temporal patterns in stock prices, improving its ability to capture diverse market trends.
- **CTTS** [74]: Combines convolutional neural networks (CNNs) with Transformer layers to capture both local feature patterns and global temporal dependencies in financial data, providing a hybrid approach to time series forecasting.
- **A2C** [75]: A deep reinforcement learning method employing parallel actor-learners and asynchronous gradient descent for policy optimization, facilitating efficient exploration and exploitation in large action spaces.
- **DDPG** [76]: A deterministic deep reinforcement learning algorithm that extends DQN by incorporating policy gradients, designed specifically for continuous action spaces, leveraging an off-policy actor-critic architecture for stable learning.
- **PPO** [77]: Optimizes policies using a clipped surrogate objective, balancing policy exploration and stability through mini-batch updates, making it robust in volatile market environments.
- **TD3** [78]: Builds on DDPG by introducing three key innovations: twin Critic networks to reduce overestimation bias, delayed updates to the Actor-network for stability, and adding noise to the policy during training to improve exploration.
- **SAC** [79]: An off-policy deep reinforcement learning approach that incorporates entropy regularization to encourage exploration, optimizing a stochastic policy for continuous actions, with dual Critic networks to improve value estimation accuracy.
- **FactorVAE** [80]: Integrates dynamic factor models with a variational autoencoder, enabling the prediction of cross-sectional stock returns by modeling latent factors that drive asset price movements.
- **AlphaStock** [52]: A hybrid model that combines deep learning and reinforcement learning with a cross-asset attention mechanism to capture the intricate relationships between assets, enhancing stock prediction accuracy by exploiting interdependencies.

Table 2: Comparing the experimental results of the models on CSI 300 and CSI 500 datasets. ARR measures the portfolio return rate of each predictive model, with higher values being better. AVol and MDD measure the investment risk of each predictive model, with lower absolute values being better. ASR, CR, and IR measure profits under unit risk, with higher values being better.

Datasets	CSI 300						CSI 500					
Model	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
BLSW	-0.076	0.113	-0.231	-0.670	-0.316	0.311	0.110	0.227	-0.155	0.485	0.710	0.446
CSM	-0.185	0.204	-0.293	-0.907	-0.631	-0.935	0.015	0.229	-0.179	0.066	0.084	0.001
LSTM	-0.214	0.175	-0.275	-1.361	-0.779	-1.492	-0.008	0.159	-0.172	-0.047	-0.044	-0.128
ALSTM	-0.216	0.164	-0.294	-1.314	-0.735	-1.461	0.016	0.162	-0.192	0.101	0.086	0.014
GRU	-0.229	0.156	-0.290	-1.469	-0.790	-1.631	-0.004	0.159	-0.193	-0.028	-0.023	-0.118
Transformer	-0.240	0.156	-0.281	-1.543	-0.855	-1.695	0.154	0.156	-0.135	0.986	1.143	0.867
TRA	-0.074	0.169	-0.222	-0.436	-0.332	-0.409	0.125	0.162	-0.145	0.776	0.866	0.657
CTTS	-0.193	0.206	-0.312	-0.937	-0.618	-0.907	-0.041	0.172	-0.239	-0.241	-0.173	-0.237
A2C	-0.207	0.092	-0.259	-2.255	-0.803	-2.490	-0.172	0.084	-0.208	-2.043	-0.826	-2.207
DDPG	-0.137	0.138	-0.240	-0.992	-0.568	-1.002	-0.128	0.082	-0.170	-1.563	-0.756	-1.639
PPO	-0.096	**0.045**	**-0.120**	-2.138	-0.800	-2.234	-0.032	**0.015**	**-0.040**	-2.041	-0.787	-2.075
TD3	-0.154	0.137	-0.252	-1.122	-0.610	-1.155	-0.123	0.135	-0.248	-0.912	-0.496	-0.909
SAC	-0.140	0.090	-0.207	-1.554	-0.676	-1.635	-0.167	0.081	-0.207	-2.057	-0.807	-2.219
FactorVAE	-0.048	0.134	-0.175	-0.335	-0.271	-0.348	0.006	0.127	-0.147	0.047	0.041	0.112
AlphaStock	-0.164	0.153	-0.245	-1.072	-0.669	-1.098	-0.017	0.148	-0.166	0.115	-0.102	-0.043
DeepPocket	-0.036	0.135	-0.175	-0.270	-0.207	-0.258	0.006	0.127	-0.148	0.050	0.043	0.115
DeepTrader	-0.122	0.147	-0.229	-0.828	-0.533	-0.876	0.055	0.168	-0.141	0.324	0.388	0.370
THGNN	-0.015	0.172	-0.152	-0.088	-0.100	-0.003	0.048	0.128	-0.141	0.375	0.340	0.432
MCI-GRU	**0.352**	0.226	-0.127	**1.559**	**2.776**	**1.526**	**0.330**	0.203	-0.198	**1.626**	**1.663**	**1.382**

- **DeepPocket** [81]: Merges graph neural networks, autoencoders, and reinforcement learning in a unified framework, focusing on managing financial portfolios by modeling the latent relationships between assets for dynamic decision-making.
- **DeepTrader** [82]: Utilizes deep reinforcement learning with graph convolutional networks to model the interrelationships between stocks, leveraging industry classifications and causal dependencies to capture both spatial and temporal market dynamics for effective portfolio management.
- **THGNN** [58]: A sophisticated temporal-heterogeneous graph neural network that integrates dynamic company relationships with Transformer encoders, featuring a two-stage attention mechanism to enhance financial time series prediction by focusing on critical temporal and structural patterns.

4.3. Experimental Results

In this section, we conduct a rigorous evaluation of the experimental results of our proposed model in comparison with several baseline models across different datasets, as illustrated in Tables 2 and 3.

On the CSI 300 dataset, traditional and deep learning models, including BLSW, CSM, LSTM, ALSTM, GRU, and Transformer, generally demonstrate subpar performance, characterized by negative ARR values and low ASR, CR, and IR metrics. For instance, while the Transformer model achieves the highest ARR among the traditional models, it still reports a negative ARR of -0.240 with a maximum drawdown (MDD) of -0.281, indicating considerable risk exposure. In the case of the CSI 500 dataset, performance improves slightly, with the Transformer model recording a positive ARR of 0.154 and an ASR of 0.986. However, these values remain notably lower than those delivered by our proposed model. Although reinforcement learning models, such as TRA, CTTS, A2C, DDPG, PPO, TD3, SAC, and FactorVAE, show marginal improvements, they continue to underperform relative to our model. For example, the PPO model reports the lowest annualized volatility (AVoL) and MDD, signaling lower risk, yet its ARR and IR metrics remain negative across both datasets, underscoring its lackluster return potential. Graph-based models, including AlphaStock, DeepPocket, DeepTrader, and THGNN, show more promising results, with THGNN, for instance, achieving an ARR of -0.015 on CSI 300 and 0.048 on CSI 500. Despite these improvements, their performance still falls short of our model.

Table 3: Comparing the experimental results of the models on S&P 500 and NASDAQ 100 datasets.

Datasets	S&P 500						NASDAQ 100					
Model	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
BLSW	0.199	0.318	-0.223	0.626	0.892	0.774	0.368	0.339	-0.222	1.086	1.658	1.194
CSM	0.099	0.250	-0.139	0.396	0.712	0.584	0.116	0.242	-0.145	0.479	0.800	0.603
LSTM	0.142	0.162	-0.178	0.877	0.798	0.929	0.247	0.176	-0.128	1.403	1.930	1.386
ALSTM	0.191	0.161	-0.150	1.186	1.273	1.115	0.201	0.192	-0.183	1.047	1.098	1.032
GRU	0.124	0.169	-0.139	0.734	0.829	1.023	0.225	0.188	-0.165	1.197	1.364	1.160
Transformer	0.135	0.159	-0.140	0.852	0.968	0.908	0.268	0.175	-0.131	1.531	2.046	1.441
TRA	0.184	0.166	-0.158	1.114	1.172	1.106	0.267	0.181	-0.144	1.475	1.854	1.427
CTTS	0.154	0.161	-0.113	0.952	1.356	0.965	0.349	0.197	-0.193	1.769	1.808	1.610
A2C	0.160	0.126	-0.084	1.267	1.907	1.244	0.109	0.134	-0.114	0.816	0.957	0.844
DDPG	0.111	0.129	-0.091	0.864	1.223	0.887	0.130	0.156	-0.131	0.832	0.994	0.863
PPO	0.020	**0.089**	**-0.067**	0.220	0.291	0.263	0.148	**0.118**	**-0.104**	1.259	1.424	1.237
TD3	0.024	0.113	-0.105	0.209	0.225	0.264	0.181	0.155	-0.160	1.169	1.130	1.156
SAC	0.140	0.111	-0.069	1.263	2.011	1.242	0.162	0.139	-0.107	1.165	1.518	1.154
FactorVAE	0.160	0.142	-0.132	1.128	1.211	1.013	0.356	0.159	-0.119	2.234	2.995	1.907
AlphaStock	0.122	0.140	-0.126	0.871	0.968	0.892	0.372	0.178	-0.134	1.781	2.776	1.869
DeepPocket	0.165	0.142	-0.126	1.165	1.311	1.045	0.346	0.157	-0.116	2.197	2.971	1.882
DeepTrader	0.295	0.180	-0.181	1.635	1.628	1.425	0.716	0.248	-0.138	2.890	5.169	2.306
THGNN	0.271	0.141	-0.094	1.921	2.871	1.778	0.644	0.204	-0.146	3.147	3.414	2.543
MCI-GRU	**0.456**	0.179	-0.129	**2.549**	**3.543**	**2.197**	**0.718**	0.220	-0.118	**3.257**	**6.091**	**2.609**

Our proposed model demonstrates a significant performance advantage, achieving the highest ARR (0.352 for CSI 300 and 0.330 for CSI 500) and exhibiting superior risk-adjusted returns, as indicated by the highest ASR (1.559 for CSI 300 and 1.626 for CSI 500), CR (2.776 for CSI 300 and 1.663 for CSI 500), and IR (1.526 for CSI 300 and 1.382 for CSI 500). Similar trends are observed in Table 3, which presents results from the S&P 500 and NASDAQ 100 datasets. Traditional models such as BLSW and CSM show moderate performance, with CSM reporting an ARR of 0.099 and an ASR of 0.396 on the S&P 500, while Transformer achieves an ARR of 0.135 and an ASR of 0.852. However, the inability of these models to effectively leverage relational data hampers their overall performance. Reinforcement learning models again demonstrate improvements, with SAC and FactorVAE achieving notable ASR values of 1.263 and 1.128, respectively, on the S&P 500. On the NASDAQ 100, FactorVAE achieves an ARR of 0.356 and the highest ASR (2.234) among reinforcement learning models. Graph-based models, particularly DeepTrader and THGNN, excel on these datasets. For example, DeepTrader achieves an ARR of 0.716 and an ASR of 2.890 on the NASDAQ 100, while THGNN records an ARR of 0.644 and an ASR of 3.147, demonstrating the effectiveness of incorporating relational information into financial models. The superiority of our MCI-GRU model is further evident, as it consistently outperforms all baseline models. It achieves the highest ARR (0.456 for S&P 500 and 0.718 for NASDAQ 100) and outstanding risk-adjusted returns, reflected in ASR values of 2.549 and 3.257, CR values of 3.543 and 6.091, and IR values of 2.197 and 2.609, respectively. These results underscore the model's effectiveness in capturing both long-term and short-term dependencies within financial data.

In summary, the experimental results highlight the critical role of integrating relational data and temporal information in stock prediction models. Our proposed model consistently surpasses traditional, deep learning, and reinforcement learning benchmarks across all datasets, achieving superior returns and enhanced risk-adjusted performance metrics. This comprehensive evaluation demonstrates the robustness and efficacy of our approach in capturing the intricate dynamics of financial markets.

4.4. Parameter Sensitivity

We conducte an in-depth analysis of the model's sensitivity to key hyperparameters, including `judge_value` (Table 4), `label_t` (Table 5), `his_t` (Table 6), `hidden_size` (Table 7), `gat_heads` (Table 8), and `num_hidden_states` (Table 9) across four benchmark datasets. This analysis is conducted to assess the robustness and stability of the model's performance under varying parameter settings. The results demonstrate that the model maintains a high degree of consistency and effectiveness across a broad range of parameter values.

Table 4: The parameter sensitivity results of parameter judge_value in dataset CSI 300, CSI 500, S&P 500, and NASDAQ 100.

Metrics	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
Datasets	CSI 300						CSI 500					
0.6	0.102	0.187	-0.142	0.546	0.720	0.650	0.107	**0.162**	-0.186	0.660	0.577	0.639
0.7	0.235	**0.183**	**-0.090**	1.282	2.623	1.201	0.212	0.167	**-0.170**	1.267	1.245	1.217
0.8	**0.352**	0.226	-0.127	**1.559**	**2.776**	**1.526**	**0.330**	0.203	-0.198	**1.626**	**1.663**	**1.382**
0.9	0.284	0.241	-0.194	1.178	1.460	1.216	0.278	0.205	-0.197	1.356	1.415	1.199
Datasets	S&P 500						NASDAQ 100					
0.6	0.286	**0.162**	-0.138	1.768	2.069	1.611	0.532	**0.219**	-0.118	2.429	4.509	2.068
0.7	0.358	0.171	**-0.111**	2.098	3.236	1.878	0.378	0.219	-0.128	1.727	2.946	1.592
0.8	**0.456**	0.179	-0.129	**2.549**	**3.543**	**2.197**	**0.718**	0.220	-0.118	**3.257**	6.091	**2.609**
0.9	0.415	0.183	-0.141	2.272	2.934	2.002	0.687	0.223	**-0.091**	3.087	**7.516**	2.504

Table 5: The parameter sensitivity results of parameter label_t in dataset CSI 300, CSI 500, S&P 500, and NASDAQ 100.

Metrics	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
Datasets	CSI 300						CSI 500					
2	0.107	**0.174**	-0.131	0.611	0.812	0.780	0.179	0.212	-0.221	0.845	0.809	0.745
5	**0.352**	0.226	**-0.127**	**1.559**	**2.776**	**1.526**	**0.330**	**0.203**	-0.198	**1.626**	**1.663**	**1.382**
8	0.319	0.220	-0.153	1.451	2.083	1.455	0.293	0.228	**-0.197**	1.286	1.487	1.228
Datasets	S&P 500						NASDAQ 100					
2	0.315	**0.167**	**-0.110**	1.884	2.854	1.720	0.449	0.230	-0.126	1.951	3.568	1.778
5	**0.456**	0.179	-0.129	**2.549**	**3.543**	**2.197**	**0.718**	**0.220**	-0.118	**3.257**	**6.091**	**2.609**
8	0.400	0.177	-0.126	2.255	3.175	1.992	0.585	0.229	**-0.117**	2.549	5.002	2.154

Judge Value Sensitivity:. The `judge_value` parameter, which governs the threshold for filtering graph relationships, exhibits a stable performance profile across all datasets. Specifically, a `judge_value` of 0.8 consistently yields the best results, with the model achieving an ARR of 0.352 and an IR of 1.526 on the CSI 300 dataset (Table 4). Performance gradually declines as the `judge_value` increases beyond this threshold, suggesting that while the model is moderately sensitive to this parameter, it retains robust performance within an optimal range.

Label Time Sensitivity:. The `label_t` parameter, which defines the forecast horizon, also demonstrates strong stability across various datasets. A prediction horizon of 5 days consistently produces the best results, particularly on the NASDAQ 100 dataset, where the model achieves an ARR of 0.718 and an IR of 2.609 (Table 5). This indicates that the model effectively captures short to medium-term market trends. The marginal variation in performance across different forecast horizons further highlights the model's adaptability and resilience in maintaining predictive accuracy.

History Length Sensitivity:. The `his_t` parameter, as shown in Table 6, representing the number of historical days considered for prediction, reveals that the model is particularly stable when using a history length of 10 days. This configuration consistently results in the highest ARR and IR across all datasets. The model's ability to effectively utilize historical data without overfitting or underfitting, even as the history length varies, underscores its robustness in learning from past patterns.

Hidden Size Sensitivity:. The hidden size, which defines the dimensionality of the model's internal representations, plays a significant role in shaping the model's performance. Our analysis identifies a hidden size of 32 (Table 7) as the optimal configuration. The model's performance remains stable across different hidden size values, demonstrating its capability to balance model complexity with predictive accuracy, and effectively handle the diverse characteristics of financial data.

Table 6: The parameter sensitivity results of parameter his_t in dataset CSI 300, CSI 500, S&P 500, and NASDAQ 100.

Metrics	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
Datasets	CSI 300						CSI 500					
6	0.162	0.216	-0.150	0.748	1.080	0.893	0.228	0.194	-0.186	1.175	1.224	1.087
8	0.272	0.210	-0.140	1.291	1.943	1.329	0.302	0.224	-0.167	1.353	1.808	1.229
10	**0.352**	0.226	-0.127	**1.559**	**2.776**	**1.526**	**0.330**	0.203	-0.198	**1.626**	1.663	**1.382**
12	0.280	0.215	-0.169	1.299	1.655	1.347	0.317	0.214	**-0.109**	1.477	**2.910**	1.376
14	0.251	**0.193**	**-0.108**	1.300	2.323	1.216	0.213	**0.180**	-0.135	1.184	1.580	1.106
Datasets	S&P 500						NASDAQ 100					
6	0.273	**0.172**	-0.125	1.588	2.186	1.494	0.524	0.229	**-0.099**	2.283	5.283	1.986
8	0.363	0.174	**-0.106**	2.085	3.424	1.866	0.547	0.227	-0.106	2.406	5.162	2.066
10	**0.456**	0.179	-0.129	**2.549**	**3.543**	**2.197**	**0.718**	**0.220**	-0.118	**3.257**	**6.091**	**2.609**
12	0.406	0.181	-0.125	2.249	3.238	1.995	0.602	0.238	-0.115	2.529	5.248	2.131
14	0.367	0.175	-0.123	2.099	2.974	1.875	0.603	0.242	-0.103	2.496	5.845	2.100

Table 7: The parameter sensitivity results of parameter hidden_size in dataset CSI 300, CSI 500, S&P 500, and NASDAQ 100.

Metrics	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
Datasets	CSI 300						CSI 500					
16	0.179	0.194	-0.126	0.924	1.423	1.023	0.271	0.203	-0.203	1.337	1.334	1.244
32	**0.352**	0.226	-0.127	**1.559**	**2.776**	**1.526**	**0.330**	0.203	-0.198	**1.626**	1.663	**1.382**
64	0.283	0.215	-0.193	1.320	1.467	1.233	0.300	0.213	-0.120	1.408	**2.510**	1.294
128	0.204	0.204	-0.210	0.999	0.970	1.102	0.179	**0.197**	**-0.098**	0.907	1.822	0.915
256	0.173	**0.180**	**-0.102**	0.963	1.692	1.001	0.192	0.218	-0.216	0.881	0.889	0.898
Datasets	S&P 500						NASDAQ 100					
16	0.411	0.175	-0.127	2.352	3.246	2.059	0.491	0.223	-0.106	2.203	4.616	1.943
32	**0.456**	0.179	-0.129	**2.549**	**3.543**	**2.197**	**0.718**	**0.220**	-0.118	**3.257**	**6.091**	**2.609**
64	0.406	0.179	-0.134	2.263	3.023	1.995	0.603	0.242	**-0.103**	2.496	5.845	2.100
128	0.396	0.177	**-0.104**	2.231	3.793	1.974	0.511	0.226	-0.128	2.263	4.002	1.996
256	0.306	**0.172**	-0.149	1.781	2.058	1.644	0.655	0.252	-0.143	2.604	4.584	2.179

GAT Heads Sensitivity:. The number of graph attention heads (`gat_heads`) also significantly impacts the model's ability to capture the complex dependencies in the data. Our experiments show that using 4 attention heads consistently results in the highest performance, with the model achieving an ARR of 0.352 and an IR of 1.526 on the CSI 300 dataset (Table 8). This stability across different settings of `gat_heads` reflects the model's robustness in learning the intricate relationships between financial assets.

Num Hidden States Sensitivity:. The number of hidden states (`num_hidden_states`) represents the market's latent dynamics. From our experiments (Table 9), we find that using 4 or 8 hidden states yields the best performance across all datasets. Specifically, for the CSI 300 dataset, setting `num_hidden_states` to 8 results in the highest ARR of 0.356 and an IR of 1.533. Similarly, for the S&P 500 and NASDAQ 100 datasets, 4 hidden states lead to superior performance, with the model achieving an ARR of 0.456 and an IR of 2.197 on the S&P 500, and an ARR of 0.718 and an IR of 2.609 on the NASDAQ 100. This suggests that a moderate number of hidden states is optimal for capturing the underlying market structures, while too few or too many hidden states can result in a decline in performance, as seen with the configurations of 2 and 16 hidden states.

Overall Observations:. Across all evaluated parameters, the model consistently exhibits a high level of stability and resilience, with minimal performance fluctuations under varying settings. This robustness suggests that the model is well-regularized and capable of maintaining strong predictive accuracy across a wide range of hyperparameter config-

Table 8: The parameter sensitivity results of parameter gat_heads in dataset CSI 300, CSI 500, S&P 500, and NASDAQ 100.

Metrics	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
Datasets	CSI 300						CSI 500					
1	0.253	**0.203**	**-0.110**	1.250	2.300	1.215	0.200	**0.195**	-0.201	1.023	0.992	0.961
2	0.285	0.210	-0.150	1.359	1.907	1.383	0.296	0.229	**-0.130**	1.290	**2.270**	1.199
4	**0.352**	0.226	-0.127	**1.559**	**2.776**	**1.526**	**0.330**	0.203	-0.198	**1.626**	1.663	**1.382**
8	0.239	0.211	-0.133	1.133	1.794	1.084	0.300	0.230	-0.198	1.305	1.519	1.218
Datasets	S&P 500						NASDAQ 100					
1	0.322	0.181	-0.133	1.781	2.414	1.640	0.686	0.235	-0.115	2.918	5.952	2.372
2	0.385	0.183	-0.147	2.102	2.623	1.876	0.666	0.239	**-0.111**	2.794	6.028	2.287
4	**0.456**	0.179	-0.129	**2.549**	3.543	2.197	**0.718**	**0.220**	-0.118	**3.257**	**6.091**	**2.609**
8	0.440	**0.162**	**-0.116**	2.718	**3.807**	**2.345**	0.588	0.226	-0.127	2.604	4.619	2.194

Table 9: The parameter sensitivity results of parameter num_hidden_states in dataset CSI 300, CSI 500, S&P 500, and NASDAQ 100.

Metrics	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
Datasets	CSI 300						CSI 500					
2	0.274	0.207	**-0.114**	1.320	2.398	1.361	0.294	0.242	-0.256	1.215	1.147	1.133
4	0.352	0.226	-0.127	1.559	2.776	1.526	0.330	0.203	-0.198	1.626	1.663	1.382
8	**0.356**	0.222	-0.125	**1.603**	**2.846**	**1.533**	**0.381**	0.202	**-0.108**	**1.888**	**3.531**	**1.658**
16	0.199	**0.199**	-0.153	0.998	1.301	1.060	0.250	**0.198**	-0.134	1.262	1.865	1.159
Datasets	S&P 500						NASDAQ 100					
2	0.420	**0.176**	-0.135	2.382	3.120	2.084	0.618	0.235	**-0.107**	2.629	5.755	2.196
4	**0.456**	0.179	-0.129	**2.549**	3.543	**2.197**	**0.718**	**0.220**	-0.118	**3.257**	**6.091**	**2.609**
8	0.435	0.232	**-0.114**	1.875	**3.803**	1.686	0.506	0.229	-0.112	2.210	4.511	1.933
16	0.341	0.182	-0.139	1.876	2.461	1.704	0.523	0.226	-0.108	2.320	4.856	2.009

urations. Such stability is critical in practical applications, where models are required to operate under varying market conditions and data distributions. The consistency of results across different datasets and parameter configurations further underscores the reliability and utility of the proposed model in financial forecasting tasks.

4.5. Ablation Study

In this section, we conduct a comprehensive ablation study to evaluate the individual contributions of various components of our model. The model is systematically divided into four distinct modules: Use Improved GRU to Capture Temporal Features (I), Use GAT to Capture Cross-sectional Features (II), Use Multi-head Cross-attention to Capture Latent State Features (III), Model Prediction and Loss Calculation Layer (IV). Detailed experimental results are presented in Tables 10 and 11.

The results presented in Tables 10 and 11 reveal several critical insights. First, the integration of the I and II resulted in a moderate improvement in performance metrics across all datasets, indicating that these components play a significant role in capturing both temporal and relational dependencies within the data. However, the addition of the III further amplified the model's performance, suggesting that the incorporation of market-wide latent states provides a more comprehensive understanding of broader market dynamics. This enhancement underscores the importance of integrating diverse levels of information to effectively model complex financial relationships.

In comparing models that incorporate the loss calculation layer, the sub-model configurations (I+II+IV) and (I+III+IV) exhibited varying degrees of improvement, with a particularly notable enhancement in the CSI 300 and CSI 500 datasets. These results suggest that the inclusion of the loss calculation layer significantly refines the model's predictions by further processing the concatenated feature vectors. Specifically, the configuration (II+III+IV) demonstrated substantial performance gains, particularly in ARR and ASR metrics, underscoring the effectiveness of combining the GAT layer, market hidden states, and an optimized loss calculation mechanism to boost predictive accuracy.

Table 10: The ablation study results in dataset CSI 300 and CSI 500.

Datasets	CSI 300						CSI 500					
Model	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
I + II	0.076	0.172	-0.214	0.441	0.354	0.488	0.151	0.178	-0.169	0.850	0.891	0.984
I + II + III	0.155	0.166	-0.170	0.934	0.911	0.957	0.195	0.219	-0.156	0.890	1.247	0.882
I + II + IV	0.210	0.201	-0.183	1.039	1.145	1.113	0.222	0.191	-0.166	1.166	1.335	1.191
I + III	-0.069	**0.160**	-0.215	-0.435	-0.322	-0.428	0.090	**0.160**	-0.168	0.560	0.534	0.548
I + III + IV	0.110	0.239	-0.199	0.459	0.554	0.551	0.192	0.243	-0.197	0.793	0.976	0.807
II + III	0.151	0.178	-0.169	0.850	0.891	0.984	0.244	0.216	**-0.154**	1.129	1.580	1.046
II + III + IV	0.287	0.211	**-0.115**	1.360	2.498	1.390	0.244	0.200	-0.187	1.223	1.304	1.150
MCI-GRU	**0.352**	0.226	-0.127	**1.559**	**2.776**	**1.526**	**0.330**	0.203	-0.198	**1.626**	**1.663**	**1.382**

Table 11: The ablation study results in dataset NASDAQ 100 and S&P 500.

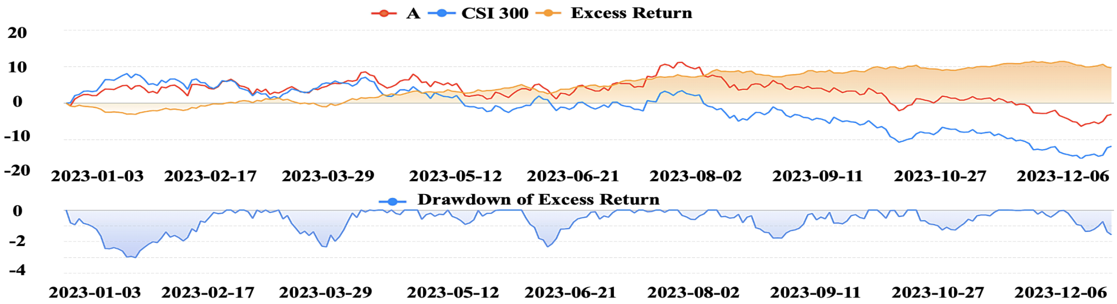
Datasets	S&P 500						NASDAQ 100					
Model	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑	ARR ↑	AVol ↓	MDD ↓	ASR ↑	CR ↑	IR ↑
I + II	0.305	0.179	**-0.120**	1.703	2.552	1.683	0.550	**0.194**	-0.101	2.841	5.461	2.483
I + II + III	0.338	0.194	-0.178	1.736	1.897	1.691	0.648	0.238	-0.112	2.721	5.807	2.248
I + II + IV	0.221	0.194	-0.155	1.143	1.428	1.240	0.444	0.195	-0.116	2.278	3.822	2.109
I + III	0.332	0.163	-0.121	2.040	2.736	1.803	0.342	0.194	-0.152	1.766	2.245	1.684
I + III + IV	0.212	**0.156**	-0.135	1.361	1.578	1.322	0.470	0.243	-0.152	1.933	3.095	1.728
II + III	0.390	0.176	-0.151	2.214	2.579	1.968	0.524	0.226	-0.109	2.320	4.771	2.026
II + III + IV	0.416	0.181	-0.157	2.296	2.647	2.018	0.564	0.235	**-0.100**	2.405	5.646	2.035
MCI-GRU	**0.456**	0.179	-0.129	**2.549**	**3.543**	**2.197**	**0.718**	0.220	-0.118	**3.257**	**6.091**	**2.609**

In conclusion, the ablation study demonstrates that the proposed model components are not only complementary but their integration substantially enhances predictive performance. This analysis underscores the critical importance of leveraging both temporal and relational information, in conjunction with latent market states, to achieve precise and reliable financial forecasting.

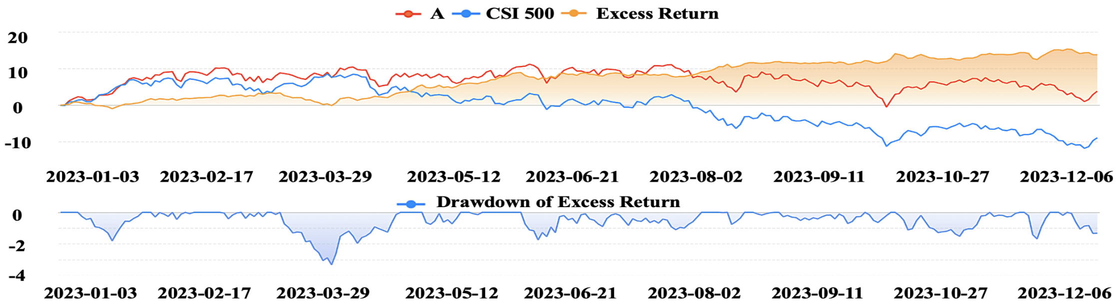
4.6. Case studies

This section offers a comprehensive explanation of the practical deployment of our model within EMoney Inc.'s real-world algorithmic trading platform, demonstrating its robust adaptability to dynamic financial environments. The model is trained on a monthly basis, generating daily predictions immediately after the close of each trading session. These predictions serve as the foundation for executing trading strategies in the initial half-hour of the next trading day. This early execution period leverages the model's predictions to capitalize on short-term market movements. The strategies we implement are tailored specifically to the CSI 300, CSI 500 and CSI 1000 stock pools, with optimization processes designed to blend the distinct characteristics of both indices, enhancing the model's overall effectiveness across diverse market conditions.

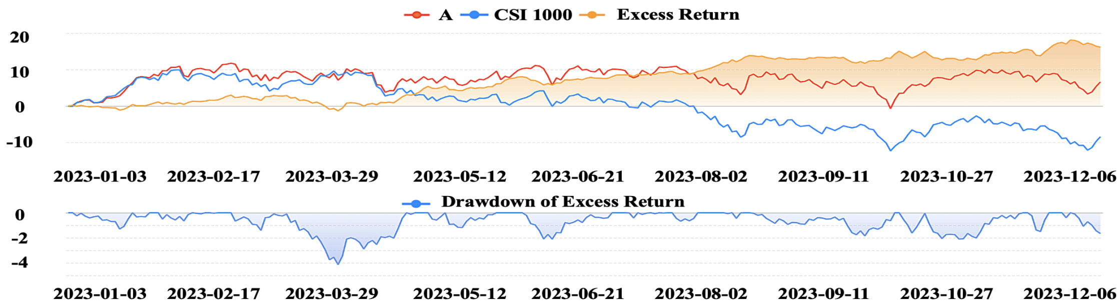
Figure 2 visualizes the performance of these strategies across different scenarios. In Figure 2(a), 2(b) and 2(c), the red curve illustrates the model's absolute returns, which reflect the actual profitability of the trades based on our model's predictions. In comparison, the blue curve shows the performance of the CSI 300, CSI 500, and CSI 1000 indices, representing the overall market return trends. The yellow curve denotes the excess returns, highlighting the model's ability to generate returns beyond the market average. Throughout a one-year period, the results consistently demonstrate that the model's strategies significantly outperform the market indices, showcasing its superior predictive power and strategic effectiveness. Moreover, the lower part of Figure 2(a), 2(b) and 2(c) provide an analysis of the excess return drawdown rate, a critical metric for assessing the model's risk management capabilities. The drawdown rate measures the extent to which excess returns decline from their peak to their lowest point, reflecting the model's ability to mitigate risk during market downturns. The model exhibits exceptional risk control, maintaining a consistently low drawdown rate, with the worst-case scenario showing a reduction of just about 5%. This low drawdown rate indicates that the model not only prioritizes return generation but also incorporates robust risk mitigation mechanisms,



(a) CSI 300 strategy backtest performance.



(b) CSI 500 strategy backtest performance.



(c) CSI 1000 strategy backtest performance.

Figure 2: The performance of the strategy backtest.

ensuring a balance between profit-seeking and risk aversion. This capability is particularly important in real-world trading environments where minimizing losses during periods of volatility is crucial for long-term success.

5. Conclusion

In this paper, we present a novel stock prediction model, MCI-GRU, which integrates a multi-head cross-attention mechanism and the improved GRU architecture to address the challenges of capturing complex temporal and relational dependencies in stock data. By replacing the reset gate in the traditional GRU with an attention mechanism, the model significantly improves its capacity to selectively utilize historical time series data. Additionally, the incorporation of the GAT enables the extraction of cross-sectional features, while the multi-head cross-attention mechanism captures latent market states that influence stock behavior. Extensive experiments conducted on both Chinese and U.S. stock market datasets demonstrate that MCI-GRU outperforms existing state-of-the-art methods across various performance metrics. Moreover, the model has been successfully implemented in a real-world fund management company, showcasing its practical applicability.

References

[1] J. Greenwood, B. D. Smith, Financial markets in development, and the development of financial markets, Journal of Economic Dynamics and Control (1997) 145–181.

[2] O. Bustos, A. Pomares-Quimbaya, Stock market movement forecast: A systematic review, Expert Systems with Applications (2020) 113464.

[3] R. Singh, S. Srivastava, Stock prediction using deep learning, Multimedia Tools and Applications (2017) 18569–18584.

[4] S. Raschka, J. Patterson, C. Nolet, Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence, Information (2020) 193.

[5] Z. Liu, Z. Zhu, J. Gao, C. Xu, Forecast methods for time series data: a survey, Ieee Access (2021) 91896–91912.

[6] Y. Gao, R. Wang, E. Zhou, Stock prediction based on optimized lstm and gru models, Scientific Programming (2021) 4055281.

[7] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, Y. Wang, Transformer in transformer, Advances in Neural Information Processing Systems (2021) 15908–15919.

[8] Y. Ding, M. Jia, Q. Miao, Y. Cao, A novel time–frequency transformer based on self–attention mechanism and its application in fault diagnosis of rolling bearings, Mechanical Systems and Signal Processing (2022) 108616.

[9] M. A. Goralski, T. K. Tan, Artificial intelligence and sustainable development, The International Journal of Management Education (2020) 100330.

[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems (2020) 4–24.

[11] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, et al., Model-based reinforcement learning: A survey, Foundations and Trends® in Machine Learning (2023) 1–118.

[12] L. Li, S. Leng, J. Yang, M. Yu, et al., Stock market autoregressive dynamics: a multinational comparative study with quantile regression, Mathematical Problems in Engineering (2016) 1–15.

[13] R. H. Shumway, D. S. Stoffer, R. H. Shumway, D. S. Stoffer, Arima models, Time Series Analysis and Its Applications: With R Examples (2017) 75–163.

[14] A. A. Ariyo, A. O. Adewumi, C. K. Ayo, Stock price prediction using the arima model, in: 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, 2014, pp. 106–112.

[15] E. S. Gardner Jr, Exponential smoothing: The state of the art, Journal of Forecasting (1985) 1–28.

[16] E. De Faria, M. P. Albuquerque, J. Gonzalez, J. Cavalcante, M. P. Albuquerque, Predicting the brazilian stock market through neural networks and adaptive exponential smoothing methods, Expert Systems with Applications (2009) 12506–12509.

[17] M. Hassan, B. Nath, Stock market forecasting using hidden markov model: a new approach, in: International Conference on Intelligent Systems Design and Engineering Applications, 2005, pp. 192–196.

[18] B. H. Juang, L. R. Rabiner, Hidden markov models for speech recognition, Technometrics (1991) 251–272.

[19] W. Huang, Y. Nakamori, S.-Y. Wang, Forecasting stock market movement direction with support vector machine, Computers & operations research (2005) 2513–2522.

[20] K.-j. Kim, Financial time series forecasting using support vector machines, Neurocomputing (2003) 307–319.

[21] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer, Knn model-based approach in classification, in: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, 2003, pp. 986–996.

[22] K. Alkhatib, H. Najadat, I. Hmeidi, M. K. A. Shatnawi, Stock price prediction using k-nearest neighbor (knn) algorithm, International Journal of Business, Humanities and Technology (2013) 32–44.

[23] F. S. D. Nugroho, T. B. Adji, S. Fauziati, Decision support system for stock trading using multiple indicators decision tree, in: International Conference on Information Technology, Computer, and Electrical Engineering, 2014, pp. 291–296.

[24] T. Kimoto, K. Asakawa, M. Yoda, M. Takeoka, Stock market prediction system with modular neural networks, in: International Joint Conference on Neural Networks, 1990, pp. 1–6.

[25] G. Zhang, B. E. Patuwo, M. Y. Hu, Forecasting with artificial neural networks:: The state of the art, International Journal of Forecasting (1998) 35–62.

[26] M. Adya, F. Collopy, How effective are neural networks at forecasting and prediction? a review and evaluation, Journal of Forecasting (1998) 481–495.

[27] H. Pan, C. Tilakaratne, J. Yearwood, Predicting australian stock market index using neural networks exploiting dynamical swings and inter-market influences, Journal of Research and Practice in Information Technology (2005) 43–55.

[28] G. S. Atsalakis, K. P. Valavanis, Surveying stock market forecasting techniques–part ii: Soft computing methods, Expert Systems with Applications (2009) 5932–5941.

[29] R. A. Kamble, Short and long term stock trend prediction using decision tree, in: 2017 International Conference on Intelligent Computing and Control Systems, 2017, pp. 1371–1375.

[30] M. Zhang, X. Jiang, Z. Fang, Y. Zeng, K. Xu, High-order hidden markov model for trend prediction in financial time series, Physica A: Statistical Mechanics and its Applications (2019) 1–12.

[31] Y. Chen, Y. Hao, A feature weighted support vector machine and k-nearest neighbor algorithm for stock market indices prediction, Expert Systems with Applications (2017) 340–355.

[32] J. Patel, S. Shah, P. Thakkar, K. Kotecha, Predicting stock market index using fusion of machine learning techniques, Expert Systems with Applications (2015) 2162–2172.

[33] R. Akita, A. Yoshihara, T. Matsubara, K. Uehara, Deep learning for stock prediction using numerical and textual information, in: International Conference on Computer and Information Science, 2016, pp. 1–6.

[34] M. E. Karim, M. Foysal, S. Das, Stock price prediction using bi-lstm and gru-based hybrid deep learning approach, in: Proceedings of Third Doctoral Symposium on Computational Intelligence, 2022, pp. 701–711.

[35] D. M. Nelson, A. C. Pereira, R. A. De Oliveira, Stock market's price movement prediction with lstm neural networks, in: International Joint Conference on Neural Networks, 2017, pp. 1419–1426.

[36] U. Gupta, V. Bhattacharjee, P. S. Bishnu, Stocknet—gru based stock index prediction, Expert Systems with Applications (2022) 117986.
[37] J. Wang, T. Sun, B. Liu, Y. Cao, H. Zhu, Clvsa: a convolutional lstm based variational sequence-to-sequence model with attention for predicting trends of financial markets, in: International Joint Conference on Artificial Intelligence, 2019, pp. 3705–3711.
[38] F. Feng, H. Chen, X. He, J. Ding, M. Sun, T.-S. Chua, Enhancing stock movement prediction with adversarial training., in: International Joint Conference on Neural Networks, 2019, pp. 5843–5849.
[39] R. E. Farmer, The stock market crash of 2008 caused the great recession: Theory and evidence, Journal of Economic Dynamics and Control (2012) 693–707.
[40] Q. He, J. Liu, S. Wang, J. Yu, The impact of covid-19 on stock markets, Economic and Political Studies (2020) 275–288.
[41] S. Sun, R. Wang, B. An, Reinforcement learning for quantitative trading, ACM Transactions on Intelligent Systems and Technology (2023) 1–29.
[42] H. Yang, X.-Y. Liu, S. Zhong, A. Walid, Deep reinforcement learning for automated stock trading: An ensemble strategy, in: Proceedings of the ACM International Conference on AI in Finance, 2020, pp. 1–8.
[43] R. Neuneier, Optimal asset allocation using adaptive dynamic programming, in: Proceedings of the International Conference on Neural Information Processing Systems, 1995, pp. 952–958.
[44] S. Carta, A. Ferreira, A. S. Podda, D. R. Recupero, A. Sanna, Multi-dqn: An ensemble of deep q-learning agents for stock market forecasting, Expert systems with Applications (2021) 113820.
[45] O. Jin, H. El-Saawy, Portfolio management using reinforcement learning, Stanford University (2016).
[46] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, Deep direct reinforcement learning for financial signal representation and trading, IEEE Transactions on Neural Networks and Learning Systems (2016) 653–664.
[47] Y. Ding, W. Liu, J. Bian, D. Zhang, T.-Y. Liu, Investor-imitator: A framework for trading knowledge extraction, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1310–1319.
[48] M. Corazza, A. Sangalli, Q-learning and sarsa: a comparison between two intelligent stochastic control approaches for financial trading, University Ca'Foscari of Venice, Dept. of Economics Research Paper Series No (2015) 1–23.
[49] R. A. de Oliveira, H. S. Ramos, D. H. Dalip, A. C. M. Pereira, A tabular sarsa-based stock market agent, in: Proceedings of the ACM International Conference on AI in Finance, 2020, pp. 1–8.
[50] K. Dabérius, E. Granat, P. Karlsson, Deep execution-value and policy based reinforcement learning for trading and beating market benchmarks, Available at SSRN 3374766 (2019).
[51] Z. Jiang, J. Liang, Cryptocurrency portfolio management with deep reinforcement learning, in: Intelligent Systems Conference, 2017, pp. 905–913.
[52] J. Wang, Y. Zhang, K. Tang, J. Wu, Z. Xiong, Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '19, 2019, pp. 1900–1908.
[53] W. Si, J. Li, P. Ding, R. Rao, A multi-objective deep reinforcement learning approach for stock index future's intraday trading, in: International Symposium on Computational Intelligence and Design, 2017, pp. 431–436.
[54] W. Zhang, Z. Chen, J. Miao, X. Liu, Research on graph neural network in stock market, Procedia Computer Science (2022) 786–792.
[55] J. Wang, S. Zhang, Y. Xiao, R. Song, A review on graph neural network methods in financial applications, Journal of Data Science (2022) 111–134.
[56] D. Cheng, F. Yang, S. Xiang, J. Liu, Financial time series forecasting with multi-modality graph neural network, Pattern Recognition (2022) 108218.
[57] W. Xu, W. Liu, C. Xu, J. Bian, J. Yin, T.-Y. Liu, Rest: Relational event-driven stock trend forecasting, in: Proceedings of the Web Conference, 2021, pp. 1–10.
[58] S. Xiang, D. Cheng, C. Shang, Y. Zhang, Y. Liang, Temporal and heterogeneous graph neural network for financial time series prediction, in: ACM International Conference on Information & Knowledge Management, 2022, pp. 3584–3593.
[59] Y. Qiao, Y. Xia, X. Li, Z. Li, Y. Ge, Higher-order graph attention network for stock selection with joint analysis, arXiv preprint arXiv:2306.15526 (2023).
[60] T. Li, Z. Liu, Y. Shen, X. Wang, H. Chen, S. Huang, Master: Market-guided stock transformer for stock price forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2024, pp. 162–170.
[61] N. Jegadeesh, S. Titman, Cross-sectional and time-series determinants of momentum returns, The Review of Financial Studies (2002) 143–157.
[62] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., Gpt-4 technical report, arXiv preprint arXiv:2303.08774 (2023).
[63] T. B. Brown, Language models are few-shot learners, arXiv preprint arXiv:2005.14165 (2020).
[64] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., Llama: Open and efficient foundation language models, arXiv preprint arXiv:2302.13971 (2023).
[65] A. Lopez-Lira, Y. Tang, Can chatgpt forecast stock price movements? return predictability and large language models, arXiv preprint arXiv:2304.07619 (2023).
[66] Z. Chen, L. N. Zheng, C. Lu, J. Yuan, D. Zhu, Chatgpt informed graph neural network for stock movement prediction, arXiv preprint arXiv:2306.03763 (2023).
[67] S. Titman, N. Jegadeesh, Returns to buying winners and selling losers: Implications for stock market efficiency, Journal of Finance (1993) 65–91.
[68] D. M. Q. Nelson, A. C. M. Pereira, R. A. de Oliveira, Stock market's price movement prediction with lstm neural networks, in: International Joint Conference on Neural Networks, 2017, pp. 1419–1426.
[69] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, G. W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: International Joint Conference on Artificial Intelligence, 2017, pp. 2627–2633.
[70] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, in: Conference on

Neural Information Processing Systems, 2014, pp. 1–9.
[71] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 1–11.
[72] Q. Ding, S. Wu, H. Sun, J. Guo, J. Guo, Hierarchical multi-scale gaussian transformer for stock movement prediction, in: International Joint Conference on Artificial Intelligence, 2020, pp. 4640–4646.
[73] H. Lin, D. Zhou, W. Liu, J. Bian, Learning multiple stock trading patterns with temporal routing adaptor and optimal transport, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2021, pp. 1017–1026.
[74] Z. Zeng, R. Kaur, S. Siddagangappa, S. Rahimi, T. Balch, M. Veloso, Financial time series forecasting using cnn and transformer, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2023, pp. 1–4.
[75] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, 2016, pp. 1928–1937.
[76] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971 (2015).
[77] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).
[78] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, 2018, pp. 1587–1596.
[79] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al., Soft actor-critic algorithms and applications, arXiv e-prints (2018).
[80] Y. Duan, L. Wang, Q. Zhang, J. Li, Factorvae: A probabilistic dynamic factor model based on variational autoencoder for predicting cross-sectional stock returns, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 4468–4476.
[81] F. Soleymani, E. Paquet, Deep graph convolutional reinforcement learning for financial portfolio management – deeppocket, Expert Systems with Applications (2021) 115–127.
[82] Z. Wang, B. Huang, S. Tu, K. Zhang, L. Xu, Deeptrader: A deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 643–650.