Multi-iTR: A Transformer-based Multi-Task Learning Framework for Stock Closing Price Prediction

Zhenjiang Chen[a], Bin Liu[a], Jun Zheng[a,*], Pei-Gen Ye[a]

[a] *School of Cyberspace Science and Technology,Beijing Institute of Technology, Beijing, China*

Abstract

In this work, we propose Multi-iTR, a framework that leverages Transformer-based multi-task learning for more accurate stock closing price prediction. Specifically, each univariate sequence in a multivariate time series is treated as a distinct token, allowing the model to capture shared and unique features across multiple stocks within the same securities market. To exploit the commonalities in stock evolution, we introduce a variable relationship mapping extraction module built on the encoder of the original Transformer architecture, which is shared among different stocks. Concurrently, we incorporate a dedicated linear projection head for each stock to address stock-specific characteristics. Furthermore, we employ a randomized sub-task training order in each epoch to reduce overfitting and improve generalization. We evaluate Multi-iTR on a randomly selected set of 32 stocks from the Chinese stock market and compare its performance against iTransformer, Transformer, LSTM, PatchTST, DLinear, and several other state-of-the-art models. Experimental results demonstrate that Multi-iTR consistently outperforms the baselines on most stocks, while ablation studies corroborate the effectiveness of each component. The source code is publicly available at: https://github.com/bitbullhorse/StockEnv.

Keywords:
Stock price prediction, Transformer, Multi-task learning, Time series forecasting, Artificial intelligence

*Corresponding author: Jun Zheng, email address: zhengjun@bit.edu.cn
Email addresses: 3120221257@bit.edu.cn (Zhenjiang Chen), 3120245862@bit.edu.cn (Bin Liu), ypgmhxy@bit.edu.cn (Pei-Gen Ye)

1. Introduction

1.1. Motivation

With the rapid expansion of global financial markets, the number of market participants has surged, the variety of financial products has broadened, and trading volumes have grown significantly. As a crucial component of these markets, stocks attract extensive attention from researchers and investors. Predicting future stock price movements is a vital challenge, as it can inform investment decisions and risk management. This goal is closely tied to the Efficient Market Hypothesis (EMH) Lu and Xu (2024), which posits that investors can infer future stock price trends by analyzing historical market information.

In traditional finance, linear models such as the AutoRegressive Integrated Moving Average (ARIMA) Sirisha et al. (2022) method are widely employed. ARIMA captures underlying patterns by combining autoregression, differencing, and moving averages. However, linear models often fail to capture the intricate interactions among multiple variables (e.g., opening price, closing price, highest price, lowest price, and trading volume), nor do they effectively model the nonlinear nature of stock price dynamics. The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model Zolfaghari and Gholami (2021); Xing et al. (2024) focuses on volatility in residual sequences, and when used together with ARIMA, helps address the phenomenon of volatility clustering in financial time series. Despite offering improved forecasts, such hybrid models remain constrained by limited adaptability to external variables and complex market factors, particularly for long-horizon predictions.

Recent advancements in deep learning have inspired the use of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks for stock price prediction. These architectures leverage their capacity to model sequential data and capture temporal dependencies. Nonetheless, they are not without limitations: RNNs suffer from vanishing gradients, hindering the learning of long-term dependencies, while LSTMs can overfit in noisy or data-scarce environments. Furthermore, effectively integrating external influences (e.g., macroeconomic indicators or sentiment data) poses additional challenges for these sequential models, limiting their overall predictive capabilities.

Transformer-based models have, therefore, garnered considerable interest in stock price prediction. Their self-attention mechanism enables the

model to focus independently on different parts of the input sequence, capturing long-range dependencies and complex nonlinear relationships inherent in stock market data. Additionally, Transformers facilitate the parallel processing of input sequences and can more flexibly incorporate diverse external signals. Although these models may incur higher computational costs—particularly in large-scale or high-frequency scenarios—their ability to capture nuanced data interactions and handle nonlinearities makes them a promising choice for robust stock price forecasting.

1.2. Contribution

Although ARIMA, RNN, LSTM, and Transformer Vaswani (2017) are widely adopted for single-stock forecasting, their performance remains constrained when tackling the more demanding multi-stock setting. To bridge this gap, this paper proposes a novel approach called **Multi-iTR**, which extends the iTransformer model Liu et al. (2023) to multi-stock price prediction. The original iTransformer uses embedding vectors of individual variable sequences from multivariate time series and employs a Transformer encoder to capture inter-variable relationships, followed by a fully connected layer for prediction. Although iTransformer exhibits strong weather, electricity, and traffic forecasting results, applying it directly to stock markets reveals two key challenges. First, training an independent iTransformer for each of the thousands of stocks in a given market is computationally expensive. Second, it overlooks the similar evolution patterns across stocks and does not exploit their shared characteristics.

To overcome these limitations, **Multi-iTR** modifies the output network of iTransformer to accommodate multi-stock prediction tasks. By adopting a multi-task learning strategy during training, the weights in the relationship extraction module are jointly updated, effectively capturing commonalities among different stocks. This design reduces computational overhead and enhances the model's capacity to extract pertinent features across multiple stocks, boosting its overall predictive accuracy.

We conduct extensive comparative experiments with cutting-edge models, including PatchTST, iTransformer, LSTM, and Transformer, to verify the effectiveness of **Multi-iTR** in forecasting stock price trends. Empirical results show that **Multi-iTR** outperforms existing time series forecasting approaches, demonstrating its robustness and superiority for stock price prediction tasks. In summary, our work significantly advances the state of the

art in multi-stock forecasting by leveraging the power of Transformer architectures and the efficiency of multi-task learning.

The main contributions of this paper are summarized as follows:

1. We propose a new Transformer-based stock prediction framework, **Multi-iTR**, and verify its effectiveness through extensive experiments.
2. By considering the correlation among multiple stocks in a securities market, we adopt a multi-task learning strategy, enhancing the model's ability to forecast stock price movements.
3. A specialized multi-task learning training method tailored to the stock market scenario is introduced, further improving the model's generalization capability.
4. We thoroughly compare with state-of-the-art time series generation models (e.g., iTransformer and PatchTST Nie et al. (2022)), highlighting the superior predictive performance of the proposed framework.

The rest of this paper is organized as follows: Section 2 surveys related work. Section 3 offers a brief overview of the computational model. Section 4 details the proposed framework and algorithms. Experimental results and analyses are presented in Section 5, and conclusions are discussed in Section 6.

2. Related Works

2.1. Time-Series Prediction Based on Transformer

Transformer-based models have garnered extensive attention in time series forecasting due to their ability to capture long-range dependencies and leverage efficient parallel computations. This is particularly appealing for financial market applications such as stock prediction, where temporal dependencies can be long-range and highly nonlinear Wang et al. (2025); Wen et al. (2022); Zhou et al. (2021); Zeng et al. (2023); Kim and Kim (2022); Zhu et al. (2023); Su and Guan (2025). For example, Zhou et al. proposed the *Informer* model, which uses sparse attention mechanisms and a generative decoder to substantially reduce computational overhead and deliver superior performance in long-sequence forecasting Zhou et al. (2021). In contrast, Zeng et al. examined the inherent complexity of Transformers for long-horizon tasks by proposing a simple linear baseline (LTSF-Linear), emphasizing more efficient architectures Zeng et al. (2023).

Several Transformer variants have been tailored for the financial domain. The MR-Transformer model Zhu et al. (2023) combines short- and long-term

temporal features, leveraging multi-resolution dependencies to improve modeling capability in stock market data. Similarly, MSDformer Su and Guan (2025) employs a multi-scale decomposition module to capture the trend and cyclical components of time series, significantly enhancing predictive accuracy and computational efficiency.

Hybrid methods integrating Transformers with other deep learning techniques also offer promising avenues. Kim et al. introduced a *Convolutional Transformer*, merging CNNs and Transformers to efficiently model multivariate time series, including features such as trading volume and macroeconomic indicators Kim and Kim (2022). Additionally, cross-modal data fusion using Transformers has been explored for financial applications, for instance, by combining satellite imagery with time series data to refine predictive accuracy Wang et al. (2025).

Despite their remarkable success, Transformer-based models still face notable challenges regarding computational complexity and scalability, especially when applied to large-scale or high-frequency financial datasets Zeng et al. (2023); Wen et al. (2022). Future work may aim to improve the models' generalization and forecasting precision by adopting more efficient architectures (e.g., sparse attention and multi-resolution methods), incorporating multimodal information (e.g., news sentiment), and developing self-supervised pretraining strategies Wang et al. (2025); Wen et al. (2022); Zhu et al. (2023). Overall, the flexibility and modeling power of Transformer-based networks holds significant promise for stock market forecasting, offering robust support for investment decisions and risk management.

2.2. Financial Prediction Based on ANN

Artificial Neural Networks (ANNs) have also seen widespread adoption in financial market forecasting. Their robust nonlinear modeling capabilities and adaptability make them a core tool for stock price prediction and risk analysis tasks. For instance, Liu et al. introduced an ensemble model combining ANN and Long Short-Term Memory (LSTM) networks, demonstrating substantial gains in predictive accuracy by leveraging LSTM's capacity for long-term dependency modeling Liu et al. (2024a). Similarly, research by Islam et al. employed window-based techniques with ANN to enhance short-term stock price predictions Hu et al. (2021).

In the context of financial risk assessment, hybrid frameworks that merge ANN with traditional models, such as the Z-Score technique, can achieve remarkably high accuracy (e.g., up to 99.40%) for evaluating the financial

health of firms Wu et al. (2022a). ANN-based solutions have also been augmented with feature selection methods and genetic algorithms to better capture the dynamic behavior of stock markets Sun et al. (2021); Hu et al. (2021). Lu et al. proposed a Transaction-Price Relationship-based Recurrent Neural Network (TRNN) to more effectively model price fluctuations in financial time series Lu and Xu (2024). Furthermore, ensemble learning frameworks (e.g., ANN+LSTM) can harness both models' strengths while mitigating their drawbacks, thereby offering improved accuracy and resilience Liu et al. (2024a).

Despite these advances, ample room remains to enhance ANNs through improved feature selection, robust model generalization, and adaptability to evolving market conditions Xiao et al. (2024). Addressing these issues could lead to further financial prediction and risk management breakthroughs.

2.3. Hard Parameter Sharing in Multi-Task Learning

Multi-task learning (MTL) aims to boost performance across multiple tasks by sharing parameters, thus reducing storage and inference overhead. In deep learning, *Hard Parameter Sharing* has emerged as a popular approach, typically implemented by sharing lower-level layers and using specialized higher-level parameters for each task Zhang et al. (2023); Vandenhende et al. (2021). This architecture is especially beneficial for scenarios where multiple related tasks must be tackled in tandem, such as predicting multiple assets in financial markets.

The Hard Parameter Sharing paradigm traces back to Caruana's multi-task learning model, which leverages shared bottom layers to extract common features while employing task-specific layers to model unique aspects of each task Vandenhende et al. (2021); Zhang et al. (2022). In dense prediction tasks for computer vision, the UberNet model showcased the effectiveness of sharing lower-level features across multiple visual tasks Vandenhende et al. (2021). Analogously, in finance, the shared layers can capture macroeconomic signals, while task-specific layers handle, for instance, different stock price trends or volatility profiles.

Nevertheless, the efficacy of Hard Parameter Sharing in Multi-Domain Learning (MDL) remains an open question, particularly when modeling distinct markets or asset classes (e.g., stocks, bonds, commodities). Zhang et al. found that a *Bottom-Specific* Hard Parameter Sharing strategy can sometimes surpass the traditional *Top-Specific* approach, as it dedicates separate bottom layers to each domain while sharing the top layers Zhang et al. (2023,

2022). This allows for more effective domain-specific feature extraction and reduced interference between tasks. In financial contexts, such nuanced partitioning is critical because disparate economic drivers often govern different asset classes.

Other studies have sought to mitigate the task-interference issue in Hard Parameter Sharing. For example, Zhang et al. introduced a task identification mechanism in sentiment analysis, employing shared layers for universal features and a specialized identification layer to diminish interference in the shared space Zhang et al. (2021). Applied to finance, such a mechanism can separate the idiosyncrasies of different assets while retaining the benefits of shared feature extraction, improving predictive accuracy for each asset class.

Optimizing Hard Parameter Sharing also involves strategic data sampling. Liu et al. proposed a "from easy to hard" *Sample-Level Data Scheduling* strategy, prioritizing simpler samples early in training to reduce task conflicts Liu et al. (2024b). In the financial domain, this could entail training on data from stable market periods before progressively introducing data from more volatile phases, smoothing gradient updates, and improving the model's generalization.

3. Background

3.1. Stock Price Prediction

Forecasting future closing prices in stock markets is critically important for institutional and individual investors. By accurately anticipating price fluctuations, traders can better determine entry and exit points, optimize portfolio allocations, and manage financial risks.

Since historical trading information can partially reveal a listed company's underlying health and operations, we utilize the past 12 days of trading data as model input and aim to predict the future stock closing prices for the next 5 and 10 days. The input data comprise the following 20 features: *closing price*, *opening price*, *highest price*, *lowest price*, *adjusted price 1*, *adjusted price 2*, *trading volume*, *trading sum*, *daily amplitude*, *full shares daily turnover ratio*, *tradable shares daily turnover ratio*, *daily return*, *daily capital appreciation*, *equal-weighted daily market return*, *tradable market value weighted daily return*, *market capitalization weighted daily return*, *equal-weighted daily market capital appreciation*, *market capitalization weighted daily capital appreciation*, *daily risk-free return*, and *price-earnings ratio*.

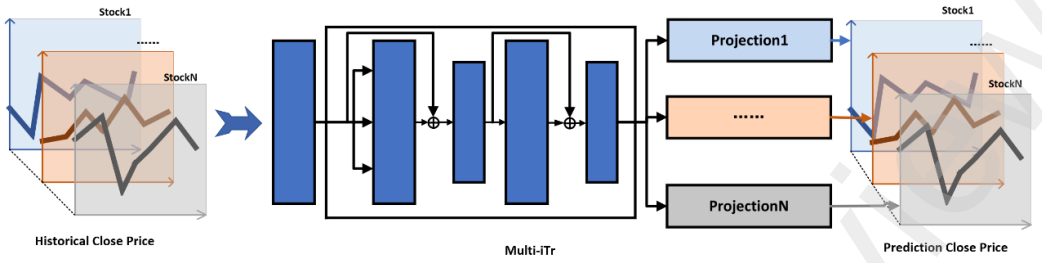

Figure 1: Overview of the Multi-iTR Framework

Formally, given the historical data matrix

$$X = \{x_1, x_2, \ldots, x_T\} \in \mathbb{R}^{T \times N}$$

and a stock identifier Str, the goal is to forecast the future closing prices

$$CP = \{cp_1, cp_2, \ldots, cp_S\} \in \mathbb{R}^{S \times 1}$$

across the next S time steps. Here, X spans the past T days, with each row containing N features (e.g., opening price, volume). The identifier Str distinguishes the stock within the broader securities market. By capturing both temporal and cross-sectional dependencies, our model aims to predict short-term and mid-term trends in closing prices robustly.

4. Methodology

In this section, we will introduce the proposed Multi-iTR model for stock price trend prediction and the training method for this model. We first provide a detailed description of the architecture of the Multi-iTR model, revealing how this architecture extracts the relationships between the various variables in the stock market trading data and predicts future stock price trends. Then, we explain how to train the Multi-iTR model to enhance its performance and generalization ability.

4.1. Multi-iTR

In this subsection, we will introduce the Multi-iTr model in detail. For convenience, we denote the stock trading data on day t as $X_{t,:}$, and the data of the n-th variable from the stock trading data across all trading days as $X_{:,n}$. The Multi-iTR model shown in Figure 2 adopts an Encoder-only Transformer architecture with multiple output heads, consisting of several projection modules, Transformer blocks, and embedding layers.

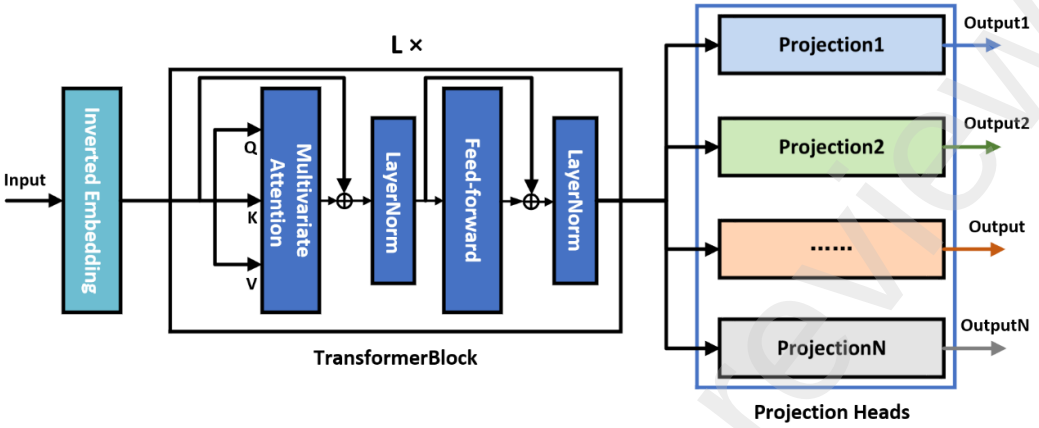

Figure 2: The architecture of Multi-iTR

4.1.1. Convert a Single Time Series into a Token

The vast majority of autoregressive stock price prediction models, including those based on Transformer and LSTM neural networks, typically treat multiple stock price trading indicators for the same stock on the same trading day as a single token and predict future tokens based on past tokens. This follows the basic paradigm of time series forecasting. However, this approach only focuses on the correlation between tokens. Specifically, traditional autoregressive prediction models based on Transformer can only learn attention maps between tokens through the self-attention mechanism, and are unable to learn attention maps between variables. Moreover, this method is also limited by the computational complexity of the self-attention mechanism (for N tokens, the complexity is $O(N^2)$), which restricts the receptive field of the model and prevents further performance improvement. A recent study Nie et al. (2022) improved the model's ability to handle long sequence tasks by dividing time series data into small patches and embedding them into a high-dimensional feature space, thereby capturing local patterns and features. This method has shown outstanding performance in many time series tasks. Furthermore, simple linear neural networks have also made significant progress in time series forecasting Zeng et al. (2023). This research suggests that the large Encoder-Decoder structure in the initial Transformer model is not necessary for time series forecasting, and that time series forecasting models consisting of simple linear neural networks can also be applied to both univariate and multivariate time series forecasting tasks. The iTransformer

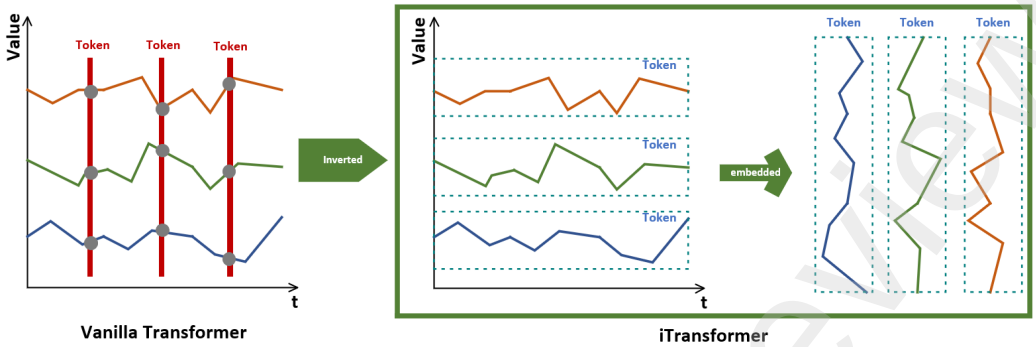

Figure 3: Time Series Vectorization

model proposed in Liu et al. (2023) combines the characteristics of linear neural networks and univariate sequence embedding. This model focuses on learning the complex relationships between statistical quantities of multivariate sequences that reflect the underlying dynamic processes, and learns these complex relationships through a multi-head self-attention mechanism. Finally, it predicts the time series using a linear neural network. Specifically, the iTransformer model treats each univariate sequence in the multivariate time series as a token, then uses the encoder of the original Transformer model to extract relationships between the variables, and finally predicts the time series using a linear neural network. The process of predicting the future sequence $Y_{:,n}$ based on the historical variable sequence $X_{:,n}$ is as follows:

$$\begin{aligned} \mathbf{h}_n^0 &= \text{invertedEmbedd}(\mathbf{X}_{:,n}), \\ \mathbf{H}^{l+1} &= \text{TRBlock}(\mathbf{H}^l), \quad l = 0, \ldots, L-1, \\ \hat{\mathbf{Y}}_{:,n} &= \text{ProjectionN}(\mathbf{h}_n^L). \end{aligned} \tag{1}$$

Here, $H = \{h_0, \ldots, h_n\} \in \mathbb{R}^{N \times D}$ contains N D-dimensional embedding vectors. Since the Embedding function Embedding : $\mathbb{R}^T \to \mathbb{R}^D$ and each Projection Head in the Projection Heads set, Projection Head : $\mathbb{R}^D \to \mathbb{R}^S$, are both linear feed-forward neural networks, it can be considered that the temporal sequence position information is implicitly encoded in the neurons of the neural networks. Therefore, there is no need for the original Transformer model's position encoder for the time sequence.

4.1.2. Transformer-Encoder

Multi-iTR uses the same encoder as the original Transformer model, but discards the decoder of the original Transformer model, reducing compu-

tational complexity. Additionally, novel attention mechanisms proposed in studies such as Zhou et al. (2021); Wu et al. (2022b); Dao et al. (2022) can be used to replace the attention module of the original Transformer model to improve computational efficiency.

The specific implementation details of the components in the Transformer Encoder of Multi-iTR are the same as those in the original Transformer model. However, the functionalities of certain components may differ.

Layer Normalization Layer normalization was proposed in Ba (2016) to improve the stability and convergence of AI model training. Traditional Transformer-based multivariate time series prediction models normalize all different scale statistics of the same timestamp. Since the statistical times of each variable may not be exactly the same, this operation introduces new noise, increases prediction errors, and may lead to excessive smoothing of the time series. However, the iTransformer (inverted Transformer) architecture used in Multi-iTR normalizes the embedding representations of the single-variable sequence, as shown in equation (2). This method normalizes each univariate time series token to a Gaussian distribution, which helps reduce errors caused by different statistical methods.

$$\text{LayerNorm}(\mathbf{H}) = \left\{ \frac{\mathbf{h}_n - \text{Mean}(\mathbf{h}_n)}{\sqrt{\text{Var}(\mathbf{h}_n)}} \;\middle|\; n = 1, \ldots, N \right\} \tag{2}$$

Feed-Forward Network Traditional Transformer-based time series prediction models typically apply the Feed-Forward Network (FFN) to tokens at the same timestamp of multivariate sequences. As mentioned earlier, due to the limitations of statistical methods, each variable at the same timestamp of a multivariate time series does not strictly originate from the same moment of the underlying complex process. This approach neglects this significant flaw and restricts tokens to represent only the state of neighboring moments rather than conveying information over longer periods, further limiting the prediction performance of the model.

However, the iTransformer (inverted Transformer) architecture employed by Multi-iTR applies FFN to tokens of univariate sequences. Leveraging the universal approximation theorem Hornik (1991), it can generate more complex embedding vectors containing richer information for prediction. Furthermore, studies such as Nie et al. (2022); Zeng et al. (2023) highlight the importance of channel independence in multivariate time series prediction tasks, while Li et al. (2023) emphasizes that temporal features extracted by

MLP for such tasks should be shared across different univariate sequences. We believe that the neurons of the MLP capture the intrinsic characteristics of the dynamic processes during training. Additionally, since each univariate in a multivariate sequence is driven by the same dynamic process, sharing the MLP module across time series can enhance the prediction performance of the model.

It is important to emphasize that we believe the dynamic evolution processes of different listed companies within the same stock market exhibit a certain degree of similarity (e.g., the same economic fundamentals, identical market regulatory measures). Thus, Multi-iTR is trained for predicting the price trends of multiple stocks within the same stock market. Consequently, we not only share MLP parameters across different variables of the same stock but also share MLP modules across different stocks. This approach enables the shared Transformer Encoder module in Multi-iTR to deeply understand the dynamic evolution processes of stocks within the stock market, which helps to further improve the model's prediction performance. Subsequent experiments also validate this advantage.

Self-Attention Mechanism In previous Transformer-based time series prediction tasks, the self-attention mechanism was typically used to model the dependencies between tokens at different timestamps. In contrast, the iTransformer (inverted Transformer) architecture adopted by Multi-iTR treats the entire sequence of a single variable as a token and leverages the self-attention mechanism to extract dependencies between variables. Specifically, for a multivariate time series $\mathbf{H} = \{\mathbf{h}_0, \ldots, \mathbf{h}_N\}$ containing N variables, the self-attention module first obtains queries, keys, and values through linear projections, $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d_k}$, where d_k represents the dimensionality of tokens after linear projection. The role of treating the entire sequence as a token in time series prediction tasks is explained in Liu et al. (2023).

It is worth emphasizing that, within the same stock market, the different variables of trading data for various stocks exhibit similar attention mappings. This is due to the fact that, to ensure market fairness, the statistical methods for trading indicators are consistent across all stocks in the same market. Furthermore, as mentioned earlier, the intrinsic evolutionary processes of stock prices share similarities. Multi-iTR fully leverages this characteristic of the stock price market. During model training and stock price prediction, the self-attention module is shared across multiple stocks. This allows the shared Transformer Encoder in Multi-iTR to fully capture the dependencies between variables in stock price prediction tasks.

4.1.3. Multi Projection Heads

Recent studies Zeng et al. (2023) have demonstrated that linear neural networks can effectively extract the intrinsic characteristics of time series and achieve remarkable results in prediction tasks. Furthermore, the success of iTransformer Liu et al. (2023) highlights the significant potential of Transformer-MLP hybrid models in time series forecasting tasks. As discussed earlier, the Transformer Encoder module of iTransformer learns the intrinsic characteristics of the dynamic evolution of multivariate time series and generates embeddings that capture variable relationships, subsequently using a linear projection layer to predict future time series.

To accommodate the complexity of the dynamic evolution of the stock market, Multi-iTR adopts the Hard Parameter Sharing method in Multi-Task Learning (MTL) Vafaeikia et al. (2020). Specifically, Multi-iTR shares the Transformer Encoder across multiple stocks while equipping each stock with a separate output head, namely, a linear projection layer. We believe that this architecture allows Multi-iTR to capture both the commonality of the dynamic evolution process in the stock market and the individuality of each stock.

In detail, during alternating training, the Transformer Encoder layer of Multi-iTR extracts fundamental features of the stock market. These features encapsulate intrinsic characteristics of the dynamic evolution of the market, fundamental rules governing the dynamic evolution of stock prices, and more. Meanwhile, the output head corresponding to each stock learns the characteristics of the price series specific to that stock. We recognize the distinction between stock-specific private features and market-wide public features, and propose that Multi-iTR effectively extracts these features to predict the future trends of stock prices.

4.2. Multi-Task Training

In the field of AI, Multi-Task Learning (MTL) aims to leverage the correlations between multiple tasks to improve the generalization performance of the model across all tasks Zhang and Yang (2021). Study Tang et al. (2020) explores how to address the two issues of negative transfer and the seesaw phenomenon in MTL. This research designed specific loss functions for each subtask and outperformed traditional models on the vast majority of subtasks.

Considering the internal complexity of each stock in the stock market, Multi-iTR also assigns an individual loss function to each stock. Further-

more, to prevent overfitting and enhance generalization performance, we randomly select the training order of all subtasks during each training epoch in Multi-iTR. We use MSE (Mean Squared Error) as the loss function during training. Additionally, during the training process, we save the model that achieves the lowest average loss across evaluation datasets for all subtasks. The overall training objective function for Multi-iTR is as follows:

$$\mathcal{L}(\Theta) = \min_{\Theta} \left(\sum_{i=1}^{M} \mathrm{MSE}_i \right), \tag{3}$$

where Θ represents the model parameters, M denotes the number of stocks, and MSE_i refers to the mean squared error of the i-th stock. The goal is to optimize the model parameters by minimizing the sum of the mean squared errors across all stocks.

In this paper, we use Algorithm1 to perform model training and evaluation.

Algorithm 1 Training and Evaluating Multi-iTR Model

Input: Model parameters Θ, dictionary of datasets *Data* for M stocks, total number of training epochs *Epochs*
Output: Trained Multi-iTR model
best_loss $\leftarrow \infty$ ▷ Initialize best loss as infinity
for *epoch* $= 1$ to *Epochs* **do**
    stock_list $\leftarrow$ random.shuffle(*Data.items*()) ▷ Shuffle the stock data to randomize training order
    for each $(stock\_name, price) \in stock\_list$ **do**
        Obtain training dataset for stock *stock_name*
        Optimize model parameters Θ on the dataset *price*
    end for
    model.eval()
    eval_losses $\leftarrow$ [] ▷ Initialize an empty list for evaluation losses
    for each $(stock\_name, price) \in Data$ **do**
        Obtain evaluation dataset for stock *stock_name*
        Evaluate the model on the dataset *price*
    end for
    avg_eval_loss $\leftarrow$ mean of *eval_losses*
    if *avg_eval_loss* $<$ *best_loss* **then**
        best_loss $\leftarrow$ *avg_eval_loss*
        best_model_wts $\leftarrow$ deepcopy of Θ
    end if
end for
$\Theta \leftarrow$ *best_model_wts*
Return: Θ

5. Experiment

5.1. Experiment Setup

We randomly selected 32 stocks from the Chinese A-share market to evaluate our proposed model. Historical data were divided into three segments:

- **Training set**: January 4, 2016 – December 30, 2022
- **Validation set**: December 31, 2022 – November 16, 2023
- **Test set**: November 17, 2023 – September 30, 2024

All data were obtained from `www.resset.com`. The experiments were conducted on an Ubuntu 20.04 platform using an Intel Core i9-11900 CPU and an NVIDIA RTX 3080Ti GPU. Each model predicts stock closing prices for the next 5 and 10 days, using the previous 12 days of trading data (e.g., opening price, closing price, volume) as input.

5.2. Experiment Results

We compared **Multi-iTR** against several baselines, including LSTM, Transformer, *iTransformer* Liu et al. (2023), *PatchTST* Nie et al. (2022), and *Dlinear* Zeng et al. (2023). Here, iTransformer and PatchTST are recent Transformer variants designed for time series prediction, while Dlinear is a linear forecasting approach known to match or exceed the performance of more complex models. For fairness, we evaluated PatchTST with both 12-day and 36-day input windows; to expand the receptive field of Dlinear, we similarly used 36 days of historical data as its input.

Table 1 and Table 2 summarize the predictive performance on the 32-stock test set, highlighting the best results in red and the second-best results with underlines. When predicting 5-day future closing prices, Multi-iTR achieves the highest accuracy on 66% of the stocks. For 10-day forecasts, it leads on 69% of the stocks. Specifically, in the 5-day prediction:

- Multi-iTR outperforms PatchTST_12 on 81% (26/32) of the stocks,
- Surpasses PatchTST_36 on 100% (32/32) of the stocks,
- Outperforms Dlinear on 66% (21/32) of the stocks.

In the 10-day forecast:

- Multi-iTR outperforms PatchTST_12 on 69% (22/32) of the stocks,
- Surpasses PatchTST_36 on 91% (29/32) of the stocks,
- Outperforms Dlinear on 69% (22/32) of the stocks.

Figures 4 and 5 depict the predicted closing price trends (for 21 of the 32 stocks) over the next 5 and 10 days, respectively. The proposed Multi-iTR model and all related code have been open-sourced on GitHub for further research and reproducibility.

Models	Multi-iTR		patchTST_36		transformer		lstm		itransformer		patchTST_12	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
600048	0.201	0.333	0.223	0.351	0.791	0.734	18.111	4.171	0.246	0.368	0.198	0.328
600030	0.386	0.379	0.437	0.389	1.434	0.946	3.557	1.647	0.425	0.402	0.397	0.380
600015	0.026	0.109	0.032	0.130	0.059	0.201	1.558	1.199	0.031	0.122	0.027	0.113
601318	1.408	0.806	1.576	0.839	7.500	2.243	38.403	5.933	1.761	0.946	1.434	0.805
600000	0.045	0.134	0.058	0.161	0.190	0.350	11.076	3.226	0.070	0.185	0.047	0.137
601088	1.477	0.931	1.644	0.998	68.792	7.510	372.089	18.934	1.957	1.081	1.521	0.952
600031	0.190	0.327	0.219	0.350	0.341	0.471	1.683	1.121	0.224	0.365	0.205	0.338
601688	0.142	0.247	0.157	0.255	12.737	3.498	5.871	2.319	0.156	0.271	0.151	0.258
601668	0.024	0.102	0.029	0.116	0.038	0.145	0.771	0.819	0.032	0.117	0.025	0.104
601328	0.014	0.088	0.015	0.091	0.055	0.178	1.013	0.860	0.017	0.099	0.014	0.088
603288	1.571	0.881	1.762	0.915	603.545	24.466	240.185	15.337	2.097	1.069	1.712	0.941
600519	1471.034	28.594	2006.619	33.016	384571.564	615.613	2129745.832	1457.447	2088.762	35.205	1485.988	29.243
600150	1.776	0.945	2.162	1.088	231.866	14.664	252.112	15.339	2.211	1.090	1.872	0.976
601818	0.006	0.053	0.007	0.056	0.181	0.405	0.306	0.538	0.007	0.060	0.006	0.052
600018	0.017	0.096	0.020	0.106	0.048	0.177	0.162	0.332	0.019	0.103	0.018	0.099
600028	0.028	0.126	0.033	0.136	0.066	0.211	1.227	1.020	0.034	0.141	0.031	0.130
600585	0.413	0.460	0.472	0.489	15.496	3.720	18.775	4.137	0.458	0.485	0.414	0.458
600372	0.154	0.300	0.165	0.318	15.158	3.844	13.156	3.574	0.181	0.323	0.163	0.303
601398	0.006	0.056	0.006	0.060	0.118	0.302	0.108	0.301	0.007	0.063	0.006	0.057
601288	0.013	0.085	0.014	0.087	0.976	0.920	0.937	0.899	0.014	0.086	0.012	0.079
601901	0.126	0.249	0.136	0.273	0.635	0.623	0.616	0.611	0.145	0.268	0.136	0.258
600050	0.016	0.085	0.017	0.086	0.054	0.192	0.417	0.612	0.018	0.095	0.017	0.089
601919	0.313	0.379	0.505	0.481	1.064	0.707	13.252	3.007	0.433	0.441	0.328	0.388
600010	0.002	0.027	0.002	0.027	0.004	0.046	0.007	0.062	0.002	0.032	0.002	0.029
601857	0.126	0.260	0.143	0.287	1.912	1.159	7.239	2.423	0.213	0.350	0.125	0.261
000001	0.068	0.187	0.089	0.220	0.150	0.295	9.309	2.970	0.081	0.209	0.076	0.196
600016	0.006	0.053	0.007	0.060	0.015	0.100	5.650	2.368	0.006	0.056	0.006	0.054
600809	72.950	6.564	76.605	6.514	4986.431	65.367	7545.664	81.323	87.112	7.258	70.060	6.304
601601	1.010	0.721	1.277	0.820	24.540	4.223	11.067	2.598	1.343	0.840	1.061	0.717
600036	0.439	0.515	0.694	0.657	13.165	2.978	8.277	2.256	0.526	0.573	0.428	0.510
600104	0.117	0.248	0.149	0.292	1.665	0.969	5.819	2.239	0.131	0.262	0.121	0.253

Table 1: Experimental Results for a Prediction Horizon of 5 Days.

Models	Multi-iTR		patchTST_36		transformer		lstm		itransformer		patchTST_12	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
600048	0.340	0.443	0.423	0.508	0.844	0.765	19.635	4.350	0.366	0.457	0.338	0.437
600030	0.451	0.457	0.514	0.483	5.153	2.044	3.727	1.700	0.520	0.496	0.483	0.466
600015	0.031	0.134	0.040	0.155	0.057	0.192	2.106	1.267	0.035	0.144	0.031	0.134
601318	1.967	1.024	2.245	1.094	196.784	13.909	65.242	7.882	2.519	1.179	1.992	1.050
600000	0.063	0.175	0.085	0.207	0.199	0.346	9.874	3.038	0.083	0.205	0.070	0.183
601088	2.403	1.210	2.739	1.272	341.944	18.137	371.473	18.933	2.779	1.343	2.357	1.204
600031	0.310	0.423	0.364	0.455	1.334	0.952	1.682	1.069	0.345	0.455	0.303	0.418
601688	0.167	0.296	0.180	0.311	0.812	0.732	8.531	2.835	0.192	0.329	0.181	0.312
601668	0.033	0.126	0.041	0.143	0.044	0.157	0.193	0.343	0.041	0.143	0.034	0.128
601328	0.021	0.110	0.027	0.122	0.213	0.335	1.042	0.884	0.025	0.122	0.021	0.109
603288	2.338	1.126	2.546	1.153	655.662	25.510	403.795	19.973	2.626	1.211	2.467	1.156
600519	2604.332	38.484	3071.716	43.453	383745.900	615.508	2133203.727	1458.872	3149.865	42.654	2725.551	39.172
600150	2.632	1.227	3.370	1.420	152.729	11.674	250.350	15.293	3.130	1.374	2.882	1.290
601818	0.008	0.069	0.010	0.081	0.014	0.094	0.023	0.123	0.010	0.077	0.008	0.070
600018	0.026	0.121	0.030	0.135	0.048	0.174	0.248	0.411	0.030	0.128	0.026	0.122
600028	0.044	0.162	0.048	0.164	0.068	0.204	0.141	0.293	0.049	0.170	0.052	0.172
600585	0.628	0.611	0.763	0.670	4.512	1.778	24.638	4.798	0.676	0.627	0.615	0.591
600372	0.250	0.393	0.246	0.397	19.253	4.346	14.803	3.800	0.272	0.412	0.240	0.386
601398	0.010	0.072	0.011	0.078	0.586	0.700	0.044	0.160	0.012	0.081	0.010	0.074
601288	0.021	0.110	0.026	0.128	0.175	0.364	0.190	0.373	0.024	0.120	0.020	0.106
601901	0.193	0.319	0.189	0.324	0.769	0.702	0.571	0.588	0.202	0.335	0.194	0.327
600050	0.023	0.109	0.022	0.107	0.060	0.198	0.390	0.589	0.027	0.119	0.023	0.109
601919	0.584	0.509	0.761	0.624	1.413	0.883	14.254	3.168	1.021	0.679	0.667	0.549
600010	0.002	0.033	0.002	0.034	0.009	0.078	0.012	0.095	0.002	0.036	0.002	0.034
601857	0.236	0.363	0.273	0.401	2.774	1.446	2.449	1.315	0.253	0.378	0.231	0.356
000001	0.124	0.254	0.153	0.298	0.529	0.599	9.609	3.020	0.132	0.274	0.130	0.257
600016	0.008	0.068	0.011	0.079	0.012	0.088	0.035	0.156	0.009	0.070	0.009	0.068
600809	127.398	9.000	154.538	9.374	5759.457	70.786	7580.655	81.621	186.227	11.154	138.000	9.383
601601	1.530	0.924	1.747	0.993	23.349	4.090	11.020	2.570	1.851	1.026	1.659	0.956
600036	0.871	0.729	1.061	0.823	17.212	3.529	8.840	2.359	1.044	0.813	0.917	0.756
600104	0.202	0.333	0.254	0.399	1.018	0.759	5.262	2.124	0.226	0.361	0.201	0.332

Table 2: Experimental Results for a Prediction Horizon of 10 Days.

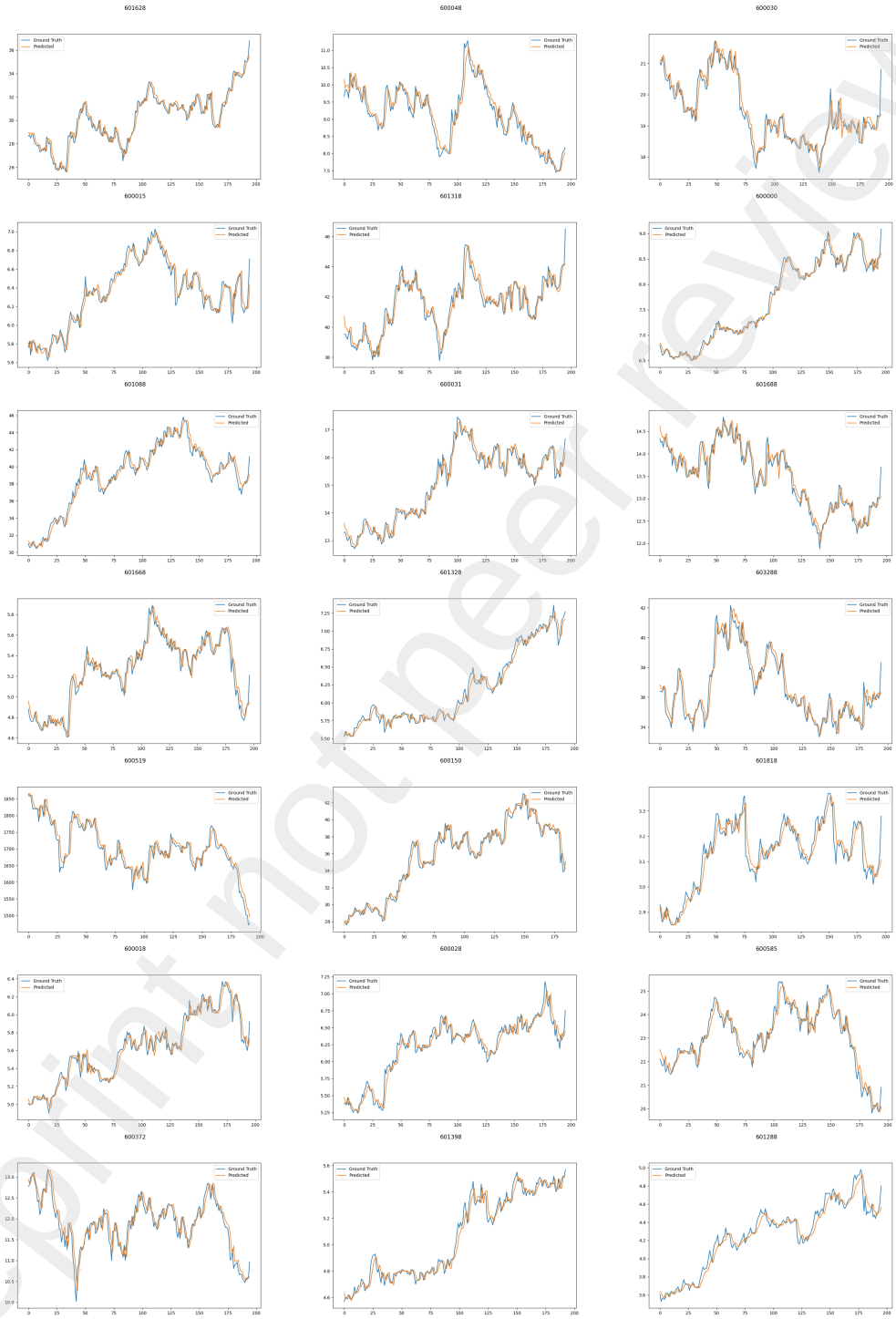

Figure 4: Results for a Prediction Horizon of 5 Days

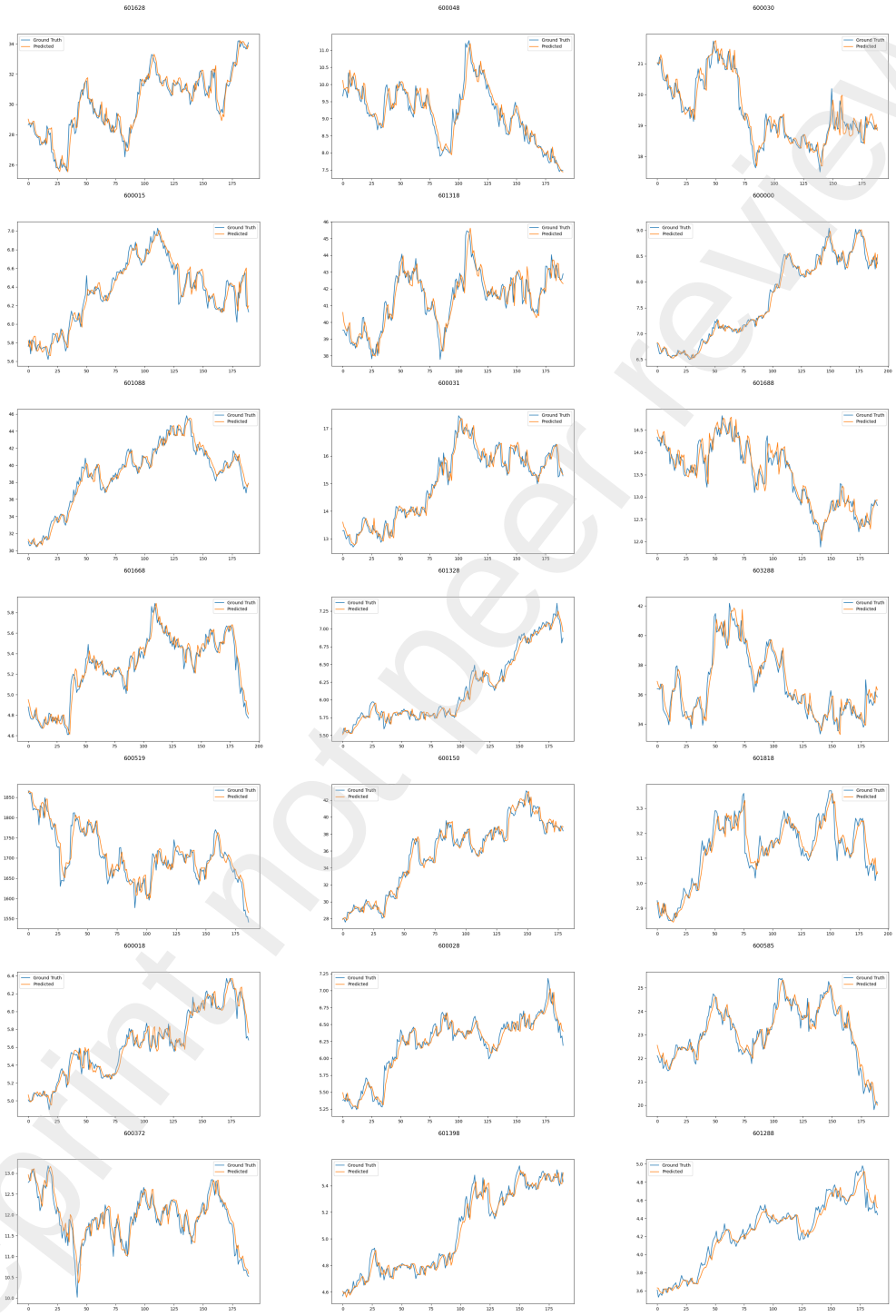

Figure 5: Results for a Prediction Horizon of 10 Days

5.3. Experiment of Method Validity

Effectiveness of Multi Task Learning To evaluate the effectiveness of multi-task learning in the Multi-iTR architecture, we randomly selected 10 stocks from a total of 32 stocks to train the Multi10-iTR, and 20 stocks to train the Multi20-iTR. For ease of comparison, the 10 selected stocks are included within the 20 selected stocks. We compared the performance of Multi-iTR, Multi10-iTR, Multi20-iTR, and iTransformer. The experimental results show that, on the randomly selected 10 datasets, when the prediction horizon is 5 days, Multi-iTR outperforms Multi10-iTR on 8 datasets, and Multi10-iTR outperforms the original iTransformer model on all 10 datasets. When the prediction horizon is 10 days, Multi-iTR outperforms Multi10-iTR on 7 datasets, and Multi20-iTR outperforms Multi10-iTR on 7 datasets, while Multi10-iTR outperforms the original iTransformer on all 10 datasets.

stock_name	**Multi-iTR**		**Multi10-iTR**		**Multi20-iTR**		**iTransformer**	
pred_len	5	10	5	10	5	10	5	10
600030	0.386	0.451	0.392	0.464	0.392	0.465	0.425	0.520
601318	1.408	1.967	-	-	1.435	2.002	1.761	2.519
600000	0.045	0.063	-	-	0.044	0.064	0.070	0.083
601688	0.142	0.167	0.143	0.170	0.143	0.166	0.156	0.192
603288	1.571	2.338	1.587	2.284	1.588	2.264	2.097	2.626
600519	1471.034	2604.332	1468.737	2676.401	1471.334	2677.349	2088.762	3149.865
600028	0.028	0.044	-	-	0.029	0.045	0.034	0.049
600585	0.413	0.628	-	-	0.410	0.654	0.458	0.676
600372	0.154	0.250	-	-	0.156	0.251	0.181	0.272
601398	0.006	0.010	-	-	0.006	0.011	0.007	0.012
601288	0.013	0.021	0.014	0.022	0.014	0.022	0.014	0.024
601901	0.126	0.193	0.126	0.192	0.126	0.189	0.145	0.202
600050	0.016	0.023	0.015	0.022	0.016	0.023	0.018	0.027
601919	0.313	0.584	0.315	0.614	0.325	0.607	0.433	1.021
600010	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
000001	0.068	0.124	0.068	0.133	0.069	0.129	0.081	0.132
600016	0.006	0.008	-	-	0.006	0.009	0.006	0.009
600809	72.950	127.398	-	-	74.731	134.512	87.112	186.227
601601	1.010	1.530	-	-	1.045	1.583	1.343	1.851
600104	0.117	0.202	-	-	0.119	0.214	0.131	0.226

Table 3: Stock performance with MSE metrics.The lowest loss in the table is marked in red, and the second lowest is underlined.

The experimental results also show that, for the selected 20 datasets, when the prediction horizon is 5 days, Multi-iTR outperforms Multi20-iTr on 18 datasets, while Multi20-iTR surpasses the original iTransformer on all 20 datasets. When the prediction horizon is 10 days, Multi-iTR outperforms

Multi20-iTR on 16 datasets, and surpasses the original iTransformer on all 20 datasets.

It is noteworthy that, when the prediction horizon is 5 days, in a randomly selected set of 10 datasets, Multi20-iTR only outperforms Multi10-iTR on 3 datasets. We believe this result may be due to the significant distributional differences across the 20 stock price datasets. However, the experimental results still show that, under the condition of equal model size, training with more data greatly improves model performance and generalization ability, which demonstrates the effectiveness and feasibility of multi-task training in the Multi-iTR framework.

The necessity of random subtask training To verify the necessity of the random subtask training order in the Multi-iTR architecture, we trained the fixed-Multi-iTR model with a fixed subtask training order for each training epoch across all 32 stocks. The experimental results show that when the prediction horizon is 5 days, Multi-iTR outperforms fixed-Multi-iTR on 30 stocks; when the prediction horizon is 10 days, Multi-iTR outperforms fixed-Multi-iTR on 24 stocks. Notably, the experimental results also indicate that fixed-Multi-iTR still outperforms most existing models on the majority of stocks.

In conclusion, the random subtask training order in Multi-iTR reduces overfitting during training and enhances the model's generalization ability, leading to improved overall performance.

stock_name	Multi-iTR		fixed_Multi-iTR		iTransformer	
pred_len	5	10	5	10	5	10
601628	1.067	1.754	1.083	1.853	1.476	2.087
600048	0.201	0.340	0.201	0.343	0.246	0.366
600030	0.386	0.451	0.390	0.456	0.425	0.520
600015	0.026	0.031	0.028	0.031	0.031	0.035
601318	1.408	1.967	1.427	2.006	1.761	2.519
600000	0.045	0.063	0.045	0.065	0.070	0.083
601088	1.477	2.403	1.524	2.472	1.957	2.779
600031	0.190	0.310	0.202	0.310	0.224	0.345
601688	0.142	0.167	0.144	0.165	0.156	0.192
601668	0.024	0.033	0.025	0.034	0.032	0.041
601328	0.014	0.021	0.014	0.022	0.017	0.025
603288	1.571	2.338	1.617	2.280	2.097	2.626
600519	1471.034	2604.332	1536.620	2566.475	2088.762	3149.865
600150	1.776	2.632	1.860	2.703	2.211	3.130
601818	0.006	0.008	0.006	0.008	0.007	0.010
600018	0.017	0.026	0.017	0.027	0.019	0.030
600028	0.028	0.044	0.028	0.045	0.034	0.049
600585	0.413	0.628	0.405	0.621	0.458	0.676
600372	0.154	0.250	0.156	0.249	0.181	0.272
601398	0.006	0.010	0.006	0.011	0.007	0.012
601288	0.013	0.021	0.013	0.021	0.014	0.024
601901	0.126	0.193	0.126	0.191	0.145	0.202
600050	0.016	0.023	0.016	0.023	0.018	0.027
601919	0.313	0.584	0.336	0.599	0.433	1.021
600010	0.002	0.002	0.002	0.002	0.002	0.002
601857	0.126	0.236	0.132	0.238	0.213	0.253
000001	0.068	0.124	0.069	0.126	0.081	0.132
600016	0.006	0.008	0.006	0.008	0.006	0.009
600809	72.950	127.398	73.401	125.904	87.112	186.227
601601	1.010	1.530	1.027	1.625	1.343	1.851
600036	0.439	0.871	0.451	0.912	0.526	1.044
600104	0.117	0.202	0.117	0.206	0.131	0.226

Table 4: Stock performance with MSE metrics.The lowest loss in the table is marked in red, and the second lowest is underlined.

6. Conclusion

The Multi-iTR proposed in this paper leverages the Transformer-Encoder to capture the relationships between various variables in the stock market and generate embedding vectors. Subsequently, a multi-head output layer based on a linear neural network is constructed to predict the future trend of stock closing prices. Experimental results demonstrate that Multi-iTR effectively extracts the intrinsic features of stock price movements in the securities market, outperforming state-of-the-art time series forecasting models, such as iTransformer, PatchTST, and Dlinear, on most stock datasets. Additionally, the results prove that our proposed method enhances the model's generalization ability and capability to capture the dynamics of the securities market. The introduction of Multi-iTR will significantly improve the ability of financial practitioners to predict and assess stock price movements, aiding investors in better stock trading, portfolio management, and risk control. Furthermore, applying advanced deep learning models in the financial sector contributes to the intelligence and automation of financial activities, reducing the reliance on human expert knowledge.

Although our proposed framework has made significant progress in stock price prediction, Multi-iTR still has certain limitations. Firstly, if there is a large disparity in the distribution of selected stock data, Multi-iTR may underperform on certain stocks compared to other models. Moreover, this paper only validates the prediction effectiveness of Multi-iTR on 32 stocks, and the model's performance has yet to be tested on a larger stock dataset. Lastly, while Multi-iTR has achieved good results on randomly selected stock data, a more reasonable stock selection mechanism is still needed to improve the model's performance further.

In future research, we will address the limitations of our current work. Specifically, we will focus on developing a more robust stock selection mechanism and validate our model on larger-scale stock datasets. Additionally, apart from predicting stock closing prices, we will explore the potential of Multi-iTR in areas such as stock liquidity prediction and stock market risk forecasting.

ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Program of China under Grant 2023YFC3305404.

References

Ba, J.L., 2016. Layer normalization. arXiv preprint arXiv:1607.06450 .

Dao, T., Fu, D., Ermon, S., Rudra, A., Ré, C., 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in Neural Information Processing Systems 35, 16344–16359.

Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. Neural networks 4, 251–257.

Hu, Z., Zhao, Y., Khushi, M., 2021. A survey of forex and stock price prediction using deep learning. Applied System Innovation 4, 9.

Kim, D.K., Kim, K., 2022. A convolutional transformer model for multivariate time series prediction. IEEE Access 10, 101319–101329.

Li, Z., Qi, S., Li, Y., Xu, Z., 2023. Revisiting long-term time series forecasting: An investigation on linear mapping. arXiv preprint arXiv:2305.10721 .

Liu, F., Guo, S., Xing, Q., Sha, X., Chen, Y., Jin, Y., Zheng, Q., Yu, C., 2024a. Application of an ann and lstm-based ensemble model for stock market prediction, in: 2024 IEEE 7th International Conference on Information Systems and Computer Aided Education (ICISCAE), IEEE. pp. 390–395.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., Long, M., 2023. itransformer: Inverted transformers are effective for time series forecasting. arXiv preprint arXiv:2310.06625 .

Liu, Z., Chai, H., Liao, Q., 2024b. Learning from easy to hard: Multi-task learning with data scheduling, in: ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 5905–5909.

Lu, M., Xu, X., 2024. Trnn: An efficient time-series recurrent neural network for stock price prediction. Information Sciences 657, 119951.

Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J., 2022. A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint arXiv:2211.14730 .

Sirisha, U.M., Belavagi, M.C., Attigeri, G., 2022. Profit prediction using arima, sarima and lstm models in time series forecasting: A comparison. IEEE Access 10, 124715–124727.

Su, G., Guan, Y., 2025. Msdformer: an autocorrelation transformer with multiscale decomposition for long-term multivariate time series forecasting. Applied Intelligence 55, 179.

Sun, J., Fujita, H., Zheng, Y., Ai, W., 2021. Multi-class financial distress prediction based on support vector machines integrated with the decomposition and fusion methods. Information Sciences 559, 153–170.

Tang, H., Liu, J., Zhao, M., Gong, X., 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations, in: Proceedings of the 14th ACM Conference on Recommender Systems, Association for Computing Machinery, New York, NY, USA. p. 269–278. URL: https://doi.org/10.1145/3383313.3412236, doi:10.1145/3383313.3412236.

Vafaeikia, P., Namdar, K., Khalvati, F., 2020. A brief review of deep multitask learning and auxiliary task learning. arXiv preprint arXiv:2007.01126 .

Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., Van Gool, L., 2021. Multi-task learning for dense prediction tasks: A survey. IEEE transactions on pattern analysis and machine intelligence 44, 3614–3633.

Vaswani, A., 2017. Attention is all you need. Advances in Neural Information Processing Systems .

Wang, K., Shan, S., Dou, W., Wei, H., Zhang, K., 2025. A cross-modal deep learning method for enhancing photovoltaic power forecasting with satellite imagery and time series data. Energy Conversion and Management 323, 119218.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., Sun, L., 2022. Transformers in time series: A survey. arXiv preprint arXiv:2202.07125 .

Wu, D., Ma, X., Olson, D.L., 2022a. Financial distress prediction using integrated z-score and multilayer perceptron neural networks. Decision Support Systems 159, 113814.

Wu, H., Wu, J., Xu, J., Wang, J., Long, M., 2022b. Flowformer: Linearizing transformers with conservation flows. arXiv preprint arXiv:2202.06258 .

Xiao, J., Wen, Z., Jiang, X., Yu, L., Wang, S., 2024. Three-stage research framework to assess and predict the financial risk of smes based on hybrid method. Decision Support Systems 177, 114090.

Xing, Y., Yan, C., Xie, C.C., 2024. Predicting nvidia's next-day stock price: A comparative analysis of lstm, mlp, arima, and arima-garch models. arXiv preprint arXiv:2405.08284 .

Zeng, A., Chen, M., Zhang, L., Xu, Q., 2023. Are transformers effective for time series forecasting?, in: Proceedings of the AAAI conference on artificial intelligence, pp. 11121–11128.

Zhang, J., Yan, K., Mo, Y., 2021. Multi-task learning for sentiment analysis with hard-sharing and task recognition mechanisms. Information 12, 207.

Zhang, L., Yang, Q., Liu, X., Guan, H., 2022. Rethinking hard-parameter sharing in multi-domain learning, in: 2022 IEEE International Conference on Multimedia and Expo (ICME), IEEE. pp. 01–06.

Zhang, L., Yang, Q., Liu, X., Guan, H., 2023. An alternative hard-parameter sharing paradigm for multi-domain learning. IEEE Access 11, 10440–10452.

Zhang, Y., Yang, Q., 2021. A survey on multi-task learning. IEEE transactions on knowledge and data engineering 34, 5586–5609.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W., 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI conference on artificial intelligence, pp. 11106–11115.

Zhu, S., Zheng, J., Ma, Q., 2023. Mr-transformer: Multiresolution transformer for multivariate time series prediction. IEEE Transactions on Neural Networks and Learning Systems .

Zolfaghari, M., Gholami, S., 2021. A hybrid approach of adaptive wavelet transform, long short-term memory and arima-garch family models for the stock index prediction. Expert Systems with Applications 182, 115149.