Discussion Papers In Economics And Business

A Search-Then-Forecast Transformer Framework
for Mid-Term Stock Price Prediction:
An Empirical Case Study on the Chinese A-Share Market

LIU JIENI

Discussion Paper 26-06

April 2026

Graduate School of Economics
The University of Osaka, Toyonaka, Osaka 560-0043, JAPAN

A Search-Then-Forecast Transformer Framework for Mid-Term Stock Price Prediction: An Empirical Case Study on the Chinese A-Share Market*

LIU JIENI†

April 2026

Abstract. This paper proposes a search-then-forecast framework for mid-term (20-trading-day) stock price forecasting and evaluates it on Chinese A-share data. The framework combines a multi-distance voting-based similar-stock search for sample augmentation, a sample-level PCA–ICA feature reconstruction, and a Transformer encoder–decoder whose decoder is initialised at inference with the single-step return at the end of the observation window rather than the conventional zero-padding. Using Luoyang Molybdenum (stock code 603993) from the SSE 180 pool as a single-stock case study, we compare six configurations—TransE, TransED (nhead = 3 and 6), BiLSTM, ARMAGARCH, and a TransED-Embed ablation—across ten observation window lengths, and introduce two baseline-referenced metrics, R^2_{hist} and $\text{MASE}_{\text{naive}}$, to address the limited interpretability of standard R^2 and MASE on non-stationary financial series. ARMAGARCH attains the lowest root mean squared error (RMSE) across all tested windows, outperforming the best deep learning model (TransE) by 1.4% to 17.0%; $\text{MASE}_{\text{naive}}$ further reveals that most deep learning models fail to surpass a random-walk naive baseline. Observation window length and model architecture exhibit a clear interaction, and a smaller internal Transformer dimension does not hurt performance. Within this single-stock case study, the findings suggest that parsimonious statistical models can match or outperform highly parameterised deep learning architectures for mid-term Chinese A-share forecasting.

Keywords: stock price forecasting; Transformer; ARMAGARCH; similar-stock search; Chinese A-share market.

JEL Classification: C22; C45; C53; G12; G17.

*The author thanks seminar participants at the Graduate School of Economics, The University of Osaka, for their valuable comments. All remaining errors are the author's own.

†Graduate School of Economics, The University of Osaka, 1-7 Machikaneyama, Toyonaka, Osaka 560-0043, Japan. Email: liu.jieni@econ.osaka-u.ac.jp.

1 Introduction

Stock price prediction has long been one of the central problems in financial research. In practice, prediction results directly inform asset allocation, risk management, and trading decisions, and improving predictive accuracy and robustness in a noisy, fast-changing market therefore remains an important question for both academia and industry. Whether such predictability is compatible with market efficiency, however, remains debated. The Efficient Market Hypothesis argues that public information is rapidly reflected in prices, so that investors can hardly obtain sustained excess returns from historical data alone (Fama, 1970). Well-documented market anomalies push back on this view: the overreaction effect—the tendency of past losers (winners) to subsequently reverse and outperform (underperform)—and the momentum effect—the tendency of recent winners to keep winning over intermediate horizons—both indicate that price deviations do not always disappear immediately (De Bondt & Thaler, 1985; Jegadeesh & Titman, 1993). Historical prices and related features may therefore still contain regularities that models can learn from.

Early approaches to stock price prediction rely on linear statistical models such as ARIMA and the GARCH family, which provide parsimonious structure but have limited capacity to capture the nonlinear dynamics of financial markets (Box, Jenkins, Reinsel, & Ljung, 2015; Siami-Namini, Tavakoli, & Namin, 2018). Deep learning models, and LSTM in particular, were introduced to address this gap (Fischer & Krauss, 2018); more recent reviews nonetheless note that recurrent architectures still struggle with long-range dependencies, and that recursive multi-step rollouts see prediction errors accumulate as the horizon grows (Zhang, Sjarif, & Ibrahim, 2024). In recent years, the Transformer architecture, with multi-head self-attention at its core, has opened a new avenue for time series forecasting (Vaswani et al., 2017; Lim, Arık, Loeff, & Pfister, 2021), and has been adopted for stock forecasting in both single-index (Ji et al., 2024) and convolution-augmented (Li, Wang, & Guo, 2024) variants. Its application to financial time series prediction, however, still faces three limitations. First, financial samples are typically limited in size and highly non-stationary (Fons, Dawson, Zeng, Keane, & Iosifidis, 2020), while Transformer performance is sensitive to data volume and model capacity. Second, the decoder input design of existing direct multi-step forecasting models is not necessarily well suited to price series. Informer, for example, pads the decoder input over the forecasting horizon with zeros at inference time (Zhou et al., 2021), so that the start of the predicted window carries no information about the most recent price level or the direction of change; Ji et al. (2024) likewise identify the decoder design as an open issue for multi-step stock forecasting and address it through a generative decoder. Third, a considerable portion of existing work still focuses on modelling a single target series (Fischer & Krauss, 2018; Siami-Namini et al., 2018), leaving cross-stock information underexploited—a gap that has motivated recent relational approaches such as MASTER (Li et al., 2024).

Motivated by these issues, this paper develops a forecasting framework targeted at mid-term (20-trading-day) stock price forecasting, using the Chinese A-share market as the research context and Luoyang Molybdenum (stock code 603993) as the single target stock in a case-study setting. The model is trained on a relative-return objective, and evaluation considers both the reconstructed closing-price sequence and the relative-return sequence output by the model. The framework proceeds in four stages. (i) Similar-stock search: for every stock in the candidate pool, we combine two normalisation schemes (Min-Max and Z-score) with three distance measures—Euclidean distance, Dynamic Time Warping (DTW), and

Shape-Based Distance (SBD)—giving $2 \times 3 = 6$ independent rankings of the pool; each ranking keeps its top-K most similar stocks, and stocks selected by at least 4 of the 6 rankings (a majority-voting threshold) are retained as the similar stocks used to augment the training sample. (ii) Feature reconstruction: the multi-dimensional features of the target and its similar stocks pass through coefficient-of-variation (CV) filtering, sample-level normalisation, and a two-stage Principal Component Analysis and Independent Component Analysis (PCA–ICA) pipeline. (iii) Forecasting: the reconstructed features are fed to a Transformer encoder–decoder, whose decoder is initialised at inference with the single-step return at the end of the observation window—projected to the model dimension through a linear layer—in place of the zero-padding used in Informer and related models, giving the forecasting window a task-relevant starting signal. (iv) Evaluation: the predicted relative returns are converted back to closing prices through an inverse transformation, and daily returns are recomputed from the predicted closing-price sequence; both dimensions are evaluated using a set of complementary metrics, among which we specifically introduce two baseline-referenced metrics, R^2_{hist} and $\text{MASE}_{\text{naive}}$, to compare the model against historical-mean and random-walk naive forecasts respectively. We choose the Chinese A-share market because retail trading dominates and trading activity is high (Tan, Zhang, & Zhang, 2024); within this market, the constituents of the SSE 180 Index are used as the similar-stock candidate pool in (i) because they offer good liquidity and relatively complete data coverage.

The contributions of this paper are fourfold. First, we propose an end-to-end search-then-forecast framework in which similar-stock search, sample-level feature reconstruction, and a Transformer encoder–decoder are integrated into a single pipeline; within it, a multi-distance voting-based similarity search combines Euclidean distance, DTW, and SBD under two normalisation schemes so that no single similarity rule dominates the retrieval, and a sample-level PCA–ICA reconstruction ensures that each observation window is normalised and projected independently. Second, we introduce a decoder-input scheme tailored to direct multi-step stock price forecasting, in which the zero-padding used in Informer and related models is replaced by the single-step return at the end of the observation window. Third, we conduct a systematic case study on Luoyang Molybdenum (stock code 603993) from the SSE 180 pool, at $L_y = 20$ and ten observation window lengths $L_x \in \{120, 140, \ldots, 300\}$, comparing ARMAGARCH against four deep learning models (TransE, TransED with nhead $= 3$ and 6, and BiLSTM) together with a TransED-Embed ablation that scales the Transformer's internal dimension. Fourth, we introduce two baseline-referenced evaluation metrics, R^2_{hist} and $\text{MASE}_{\text{naive}}$, which make the comparison against a historical-mean benchmark and a random-walk benchmark explicit. Both are formally defined in Section 4.7, and $\text{MASE}_{\text{naive}}$ directly reveals that most deep learning models fail to beat the random-walk benchmark—a phenomenon that RMSE alone cannot surface. The single-target-stock scope is a deliberate case-study choice and is explicitly discussed as a limitation in Section 6.5.

The remainder of this paper is organised as follows. Section 2 reviews related work. Section 3 details the proposed method. Section 4 describes the experimental setup. Section 5 reports the experimental results. Section 6 discusses the findings. Section 7 concludes the paper and outlines future research directions.

2 Related Work

2.1 Evolution of Time Series Forecasting Methods

Time series forecasting methods have progressed through statistical models, machine learning, and deep learning. Classical statistical models, represented by ARIMA (Autoregressive Integrated Moving Average) and the ARCH (Autoregressive Conditional Heteroskedasticity) / GARCH (Generalised Autoregressive Conditional Heteroskedasticity) family, offer a clear structure and few parameters for modelling linear series and financial volatility clustering (Box et al., 2015; Engle, 1982; Bollerslev, 1986), but have limited capacity to capture nonlinear relationships.

In the machine learning phase, support vector machines and random forests improved nonlinear fitting through kernel functions and ensembles (Kim, 2003; Huang, Nakamori, & Wang, 2005; Breiman, 2001; Ballings, Van den Poel, Hespeels, & Gryp, 2015). The review by Kontopoulou, Panagopoulos, Kakkos, and Matsopoulos (2023) reports that, when data are sufficient, machine learning methods generally achieve lower errors than ARIMA, and that hybrid approaches are more stable.

The deep learning phase began with recurrent neural networks. Among them, Long Short-Term Memory (LSTM), with its gating structure, has become the most widely used architecture for financial sequence modelling (Hochreiter & Schmidhuber, 1997; Fischer & Krauss, 2018; Siami-Namini et al., 2018; Sezer, Gudelek, & Ozbayoglu, 2020). However, the sequential computation inherent in recurrent structures does not parallelise well, and modelling long-range dependencies suffers from information decay. These limitations have driven the rise of self-attention and the Transformer architecture. Against this background, we adopt a Transformer encoder–decoder as the forecasting model. Alongside the choice of predictor, financial prediction pipelines also have a long tradition of reducing feature redundancy before modelling. Principal Component Analysis (PCA) and Independent Component Analysis (ICA) are the two most widely used tools for this purpose, and recent work by Sarıkoç and Celik (2025) shows that stacking them—PCA for denoising followed by ICA for feature extraction—before an LSTM delivers stronger performance than either stage alone on the S&P 500. We draw on this two-stage design when constructing the feature reconstruction module in Section 3.3.

2.2 Similar Stock Search in Financial Prediction

Once a Transformer has been chosen as the forecasting model, a related issue concerns the source of training samples. Relying on the historical series of a single stock alone has clear limitations: first, the training samples a single target can provide are limited; second, cross-stock co-movement in the market is left unexploited. Anomalies such as the momentum effect and overreaction suggest that regularities in historical trajectories persist over certain periods, so stocks with trajectories similar to that of the target may carry useful auxiliary signals. Building on this observation, a growing number of studies have started to incorporate information from similar stocks into forecasting models.

Identifying similar stocks typically relies on pairwise similarity measures defined on historical price or return series. The Pearson correlation coefficient measures linear co-movement between returns. DTW allows elastic alignment along the time axis and is suitable for capturing series with small time shifts but similar shapes. Paparrizos and Gravano (2015) proposed the Shape-Based Distance (SBD),

which measures overall shape similarity through a distance based on normalised cross-correlation and is insensitive to amplitude scaling. Li, Zhu, Shen, and Angelova (2023) further proposed Logistic-weighted DTW (LWDTW) for the financial setting, assigning different weights to price movements of different magnitudes so that the distance calculation better reflects practical concerns in price changes.

Empirical studies show that using information from similar stocks helps improve predictive performance. Wang, Yang, and Liu (2021) grouped stocks by shape-based similarity and combined the grouping with the Hierarchical Temporal Memory (HTM) model, obtaining results better than those from single-stock information alone in short-term prediction tasks. Li et al. (2023) combined similarity-based training-sample augmentation with deep learning predictors such as LSTM, Recurrent Neural Network, and Gated Recurrent Unit, and found that the augmentation step further reduced prediction errors. Similar-stock information is useful for two reasons. First, when the sample is small, the historical data of similar stocks can supplement the limited training samples available from the target stock. Second, similar stocks are often affected by the same industry and macroeconomic factors, and the shared signals they carry are informative for predicting the target.

These studies provide direct inspiration for this paper, but two aspects can be pushed further. First, different distance measures capture different similarity dimensions: Euclidean distance focuses on amplitude and level, DTW focuses on shape consistency under local misalignment, and SBD emphasises amplitude-independent overall contour. Relying on a single measure can easily miss certain similarity patterns. Unlike the design of Li et al. (2023), which uses a single LWDTW measure combined with recurrent predictors, we improve search robustness through a voting scheme across three distances evaluated under two normalisation schemes. Second, existing work mostly combines similar-stock search or clustering with classical machine learning or LSTM, and its systematic combination with a Transformer encoder–decoder is still rare. We use the self-attention mechanism to model dependencies across multiple stock sequences, filling this gap.

2.3 Transformer in Financial Time Series Forecasting

The Transformer, proposed by Vaswani et al. (2017), is centred on a self-attention mechanism that directly models dependencies between different positions in a sequence and supports parallel computation. Once introduced into time series forecasting, improvements to the Transformer have developed mainly along two paths.

One path focuses on the computational efficiency of attention and the ability to model long sequences. Informer uses ProbSparse attention to reduce the computational complexity from $O(L^2)$ to $O(L \log L)$, and introduces a generative-style decoder that outputs the entire forecasting sequence in parallel within a single forward pass (Zhou et al., 2021). At inference, this decoder pads the forecasting horizon with zeros. For price series with marked short-term momentum, such zero-padding means that the start of the forecasting window carries no information about the recent price level or direction of change, and the model must rely entirely on the encoder output to infer a reasonable starting point. Autoformer improves long-horizon forecasting through series decomposition and an auto-correlation mechanism (Wu, Xu, Wang, & Long, 2021), while FEDformer models series from a frequency-domain perspective (Zhou, Ma, Wen, Wang, Sun, & Jin, 2022). The other path focuses on gating structures and interpretability. The Temporal Fusion Transformer combines gated residual networks, local sequence processing, and multi-head attention for

interpretable multi-step forecasting (Lim et al., 2021).

Within financial applications, some studies still focus on a single target series, while others directly exploit cross-stock relationships. Li, Bao, Harimoto, Chen, Xu, and Su (2020) used graph networks to model inter-stock relations such as industry and supply chain, applied to overnight price movement prediction. Li et al. (2024) proposed MASTER, which alternates intra-stock temporal feature aggregation and cross-stock relational feature aggregation under a Transformer framework, and introduces a market-information-guided gating mechanism for dynamic feature selection. This line of work shows that cross-stock dependencies can provide additional predictive information for the target. Unlike these approaches, which rely on graph structures or gating networks to combine contemporaneous cross-stock features within each input sample, we use similar-stock search as an offline sample-augmentation step: the retrieved similar-stock series enter the training set as additional independent samples, and the Transformer is asked to model each stock's trajectory separately under a shared set of parameters. The underlying hypothesis is that, by learning to forecast both the target and stocks with similar trajectories, the model acquires representations that transfer to the target stock's unseen test windows. This design imposes weaker requirements on data and model architecture, which makes systematic experiments within a single market more practical.

In summary, several directions in Transformer-based financial forecasting remain worth exploring. The tension between the limited scale of financial samples and the parameter count of Transformers calls for data augmentation strategies, and we introduce similar-stock search for this purpose. The decoder input design in direct multi-step forecasting directly affects the quality of the starting signal for the forecasting window, and discussion of this aspect is still limited: Ji et al. (2024) replace the iterative decoder with a single-pass generative one for stock index forecasting, and we pursue a complementary direction by changing what the decoder is initialised with rather than how it generates. In addition, the above improvements have not been systematically verified on the Chinese A-share market, particularly under the combination of multi-distance similar-stock search and direct multi-step forecasting. Our experiments aim to supplement evidence in this direction.

3 Method

This section introduces the proposed forecasting framework, which consists of three components: similar-stock search, feature reconstruction, and direct multi-step forecasting. Similar-stock search is used to augment the training sample. Feature reconstruction processes the raw input. The forecasting module produces predictions for the future window based on the reconstructed sequence. Figure 1 gives an overview of the pipeline. Evaluation metrics tailored to direct multi-step forecasting are formally defined in the experimental part of Section 4.7.

3.1 Notation and Problem Setup

3.1.1 Notation

We denote by L_x and L_y the look-back window length (observation window) and the forecasting horizon, respectively. L_x is the number of historical time steps used to extract input information, and L_y is the

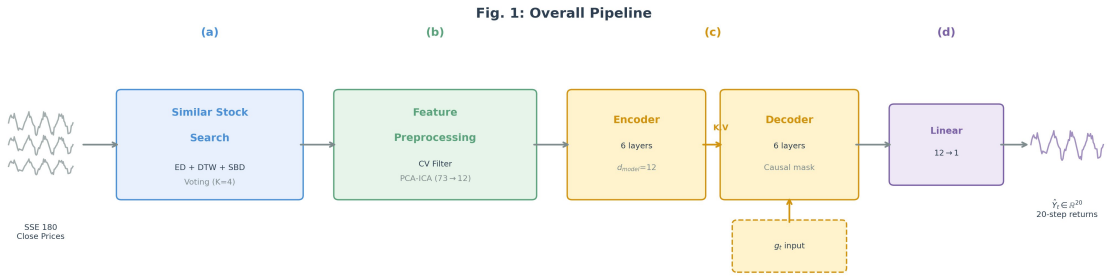

Figure 1. Overall Pipeline. (a) Similar-stock search. (b) Feature reconstruction, consisting of coefficient-of-variation (CV) filtering, sample-level normalisation, and PCA–ICA reduction. (c) Transformer encoder–decoder. (d) 20-step relative-return output.

number of future time steps produced in a single prediction. L_y is fixed at 20 trading days, corresponding to a mid-term forecasting horizon of roughly one month. L_x serves as the experimental variable and is systematically varied in the experimental section; its tested range and the rationale are given in Section 4.

Let d_{raw} denote the original candidate feature dimension and d_{in} the input dimension after feature reconstruction. The raw candidate features include price, trading volume, and commonly used technical indicators, totalling d_{raw} dimensions; the complete list is given in Section 4.1. For any time t, the raw observation window is denoted $X_t^{\text{raw}} \in \mathbb{R}^{L_x \times d_{\text{raw}}}$, and the reconstructed input matrix is denoted $Z_t \in \mathbb{R}^{L_x \times d_{\text{in}}}$. Let p_t denote the closing price of the target stock at time t. The future target sequence is denoted Y_t, and the model prediction is denoted $\hat{Y}_t$.

3.1.2 Multi-step Forecasting Task

We conduct multi-step forecasting under a rolling window framework. At each time t, the model uses the historical window Z_t to predict a target sequence of length L_y. This mapping is written as

$$\hat{Y}_t = f_\theta(Z_t),$$

where f_θ is the mapping function to be learned and θ denotes the model parameters. We use the relative return as the prediction target. The return at the j-th future step relative to the end of the observation window t is defined as

$$r_{t+j|t} = \frac{p_{t+j}}{p_t} - 1, \qquad j = 1, 2, \ldots, L_y.$$

The model outputs L_y relative return predictions directly. At the evaluation stage, the predicted relative returns $\hat{r}_{t+j|t}$ are converted to closing prices through the inverse transformation $\hat{p}_{t+j} = (\hat{r}_{t+j|t} + 1)\, p_t$. Model performance can therefore be evaluated in two complementary dimensions: on the reconstructed closing-price sequence and directly on the relative-return sequence output by the model. The subsequent discussion of the decoder input design is based on the relative-return formulation.

3.1.3 Direct Multi-step Forecasting Paradigm

Two common strategies exist for multi-step forecasting: recursive forecasting and direct multi-step forecasting (DMS). Recursive forecasting generates one step at a time and feeds the output back into the

model to compute the next step. DMS produces the entire forecasting window in a single pass.

We adopt DMS. In recursive forecasting, the prediction error at each step propagates to subsequent steps. The bias–variance analysis of Taieb and Atiya (2016) shows that, on noisy and non-stationary series, the error variance of recursive strategies grows rapidly with the forecasting horizon, while DMS effectively controls this accumulation by modelling each horizon independently. This analysis is consistent with empirical observations on volatile financial time series. DMS also has a practical advantage: it does not require feeding intermediate predictions back at inference, but directly models the full future window. This choice aligns with the task setting of this paper, which concerns a complete future window rather than any single future time point.

3.2 Similar-Stock Search and Sample Augmentation

For a single target stock, modelling with its own historical series alone often cannot provide enough training samples, and the problem becomes more pronounced as model complexity increases. To reduce the overfitting risk caused by sample scarcity, we introduce similar-stock search at the training stage, selecting stocks from the SSE 180 constituents whose price trajectories are close to that of the target stock and using their historical data to augment the training sample. Within the same market environment, different stocks may be affected by similar macroeconomic factors, industry events, or market sentiments, and may display similar volatility patterns or shapes in certain periods. Their historical series can therefore provide useful auxiliary signals for predicting the target stock.

3.2.1 Similarity Measures

To select stocks whose price trajectories are close to that of the target, we jointly use three distance measures—Euclidean distance (ED), Dynamic Time Warping (DTW), and Shape-Based Distance (SBD)—each computed under two normalisation schemes (Min-Max and Z-score), giving the six rankings that feed into the voting stage in Section 3.2.2. The three distances are defined below on a pair of length-L_s series, where L_s denotes the length of the price series used for similar-stock search (fixed at 250 trading days in our experiments; see Section 4.1). This L_s is deliberately distinguished from the observation window length L_x used by the forecasting model. Let x_{target} denote the target series and x_i a candidate series.

Euclidean distance directly compares the two series at aligned time positions:

$$d_{\text{ED}}(x_{\text{target}}, x_i) = \sqrt{\sum_{k=1}^{L_s} (x_{\text{target},k} - x_{i,k})^2}.$$

ED is sensitive to series that are close in both overall level and fluctuation amplitude, but it requires strict pointwise alignment.

Dynamic Time Warping allows the two series to stretch or compress locally along the time axis, and is therefore better suited to pairs with small lead–lag offsets but otherwise similar trajectories. Let $c(k,l)$ denote the cumulative alignment cost between the prefixes $x_{\text{target}}[1{:}k]$ and $x_i[1{:}l]$. The cost satisfies the

recursion

$$c(k,l) = |x_{\text{target},k} - x_{i,l}| + \min\{c(k-1,l-1),\ c(k-1,l),\ c(k,l-1)\},$$

with boundary conditions $c(0,0) = 0$ and $c(k,0) = c(0,l) = \infty$ for $k,l > 0$. The final DTW distance is $d_{\text{DTW}}(x_{\text{target}}, x_i) = c(L_s, L_s)$. Compared with ED, DTW does not require pointwise alignment and is therefore more robust to local time shifts.

Shape-Based Distance, proposed by Paparrizos and Gravano (2015), measures shape similarity through normalised cross-correlation. Let $R_s(x_{\text{target}}, x_i)$ denote the cross-correlation at lag s. Then

$$\text{SBD}(x_{\text{target}}, x_i) = 1 - \max_s \frac{R_s(x_{\text{target}}, x_i)}{\|x_{\text{target}}\|_2 \cdot \|x_i\|_2}.$$

SBD takes values in $[0,2]$, with smaller values indicating closer shape. Because the denominator normalises the amplitude, SBD is insensitive to scale and focuses on contour features such as peaks and troughs.

The three measures capture different similarity dimensions. Combining them reduces the bias that a single distance may introduce.

3.2.2 Similar-Stock Selection and Augmented Sample Pool

In the concrete implementation, we first apply both Min-Max and Z-score normalisation to the closing price series of the most recent 250 trading days for every stock, reducing the dependence on any single numerical scale. On each normalised version, we compute the three distances (ED, DTW, SBD) between the target stock and each candidate stock. This yields 2 normalisations $\times$ 3 distances $=$ 6 independent rankings, and the top-5 stocks in each ranking (by ascending distance) form the Top-K candidate list for that ranking. At the voting stage, each candidate is scored by the number of times it appears in the six Top-K lists, and a threshold of $4/6$ (i.e., selected by at least four rankings—a majority-voting criterion) is used to retain the final similar stocks. Let M denote the number of similar stocks retained. Together with the target stock, they form an augmented sample pool of size $M+1$. Figure 2 illustrates this search procedure.

On this basis, we construct training samples with a stride-1 sliding window. For each stock in the augmented sample pool, we extract the raw observation window X_t^{raw} and its corresponding target sequence Y_t. The data are split strictly in chronological order: the last continuous segment of samples along the time axis is reserved as the test set, the segment immediately preceding it as the validation set, and the remainder as the training set. This split preserves the temporal order of the series and avoids information leakage that would arise from overlaps between adjacent sliding windows.

3.3 Feature Reconstruction Framework

After obtaining the augmented sample pool, the raw input may still suffer from three issues: a relatively high feature dimension, redundancy across features, and marked scale differences across time windows. In addition, longer observation windows contain more historical information but may also introduce noise weakly related to the current prediction. Feeding the raw features directly into the forecasting model

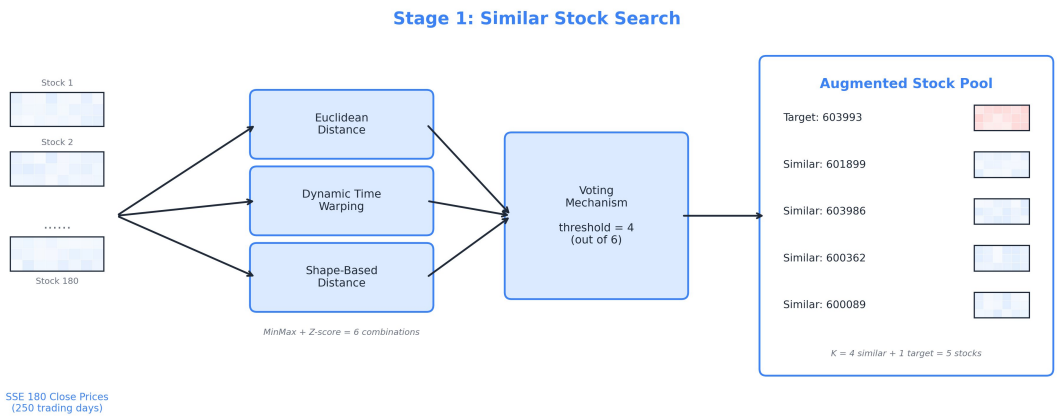

Figure 2. Similar-Stock Search Procedure. Each candidate stock in the pool is evaluated against the target under two normalisation schemes (Min-Max and Z-score) and three distance measures (ED, DTW, and SBD), giving $2 \times 3 = 6$ independent rankings. Each ranking keeps its top five stocks as candidates, producing six candidate lists. Stocks selected by at least four of the six lists—a majority-voting threshold ($\geq 4/6$)—are retained as the M final similar stocks, which together with the target stock form the augmented sample pool. In the experiment of this paper, $M = 4$, and the selected similar stocks for the target Luoyang Molybdenum (603993) are Zijin Mining (601899), GigaDevice Semiconductor (603986), Jiangxi Copper (600362), and TBEA (600089).

forces the model to handle temporal dependencies together with noise and redundancy, making training more difficult. For this reason, we build a feature reconstruction framework consisting of feature filtering, local normalisation, and PCA–ICA reconstruction, which transforms the raw observation window X_t^{raw} into an input matrix Z_t more suitable for downstream forecasting.

3.3.1 Feature Filtering

Feature filtering proceeds in two steps, addressing feature degeneracy, target relevance, and inter-feature collinearity in turn.

The first step is filtering based on the coefficient of variation (CV). Some features vary very little within the observation window and carry almost no discriminative information. For each sample window, we compute the coefficient of variation of each feature. If a feature's CV falls below a threshold across all sample windows, it is treated as a degenerate feature and removed. This step mainly eliminates constant or near-constant features.

The second step is a two-round filtering based on the Pearson correlation coefficient. Let $x^{(i)}$ denote the i-th feature series and y denote the prediction target. The Pearson correlation between them is

$$r(x^{(i)}, y) = \frac{\sum_{t=1}^{T} (x_t^{(i)} - \bar{x}^{(i)})(y_t - \bar{y})}{\sqrt{\sum_{t=1}^{T} (x_t^{(i)} - \bar{x}^{(i)})^2} \cdot \sqrt{\sum_{t=1}^{T} (y_t - \bar{y})^2}}.$$

We first keep candidate features that satisfy $|r(x^{(i)}, y)| > \tau_1$, removing input items weakly related to the target. Among the retained features, we then check pairwise correlations: if two features are correlated above the threshold τ_2, we keep the one with higher target correlation and remove the other, reducing collinearity in downstream modelling. The specific values of τ_1 and τ_2 are given in Section 4.

3.3.2 *Local Normalisation*

After feature filtering, noticeable scale differences may still exist across sample windows. Financial time series are often non-stationary, with mean levels and volatility differing substantially across periods. A single global normalisation would weaken local variation patterns within each window.

For this reason, we apply a sample-level two-step local normalisation. First, for any raw observation window X_t^{raw} of length L_x, we compute the local minimum $\min_{\text{local}}$ and local maximum $\max_{\text{local}}$ of each feature dimension within the window, and linearly scale the feature to $[0,1]$:

$$X_t' = \frac{X_t^{\text{raw}} - \min_{\text{local}}}{\max_{\text{local}} - \min_{\text{local}} + \varepsilon}.$$

Second, we subtract the within-window mean μ_{local} from the result of the first step, giving a zero-centred locally normalised sequence:

$$\tilde{X}_t = X_t' - \mu_{\text{local}},$$

where ε is a small constant used to avoid a zero denominator in the first step. The combination of the two steps serves two purposes. First, Min-Max scaling unifies the scale of the features across dimensions, preventing any high-magnitude feature from dominating downstream modelling. Second, zero-mean centring within $[0,1]$ makes the features at each time position distributed symmetrically around the within-window average, which helps the attention mechanism extract relative variation patterns within the local window. Zero-mean centring also provides the centred input required by PCA.

3.3.3 *Two-stage PCA–ICA Reconstruction*

After feature filtering and local normalisation, redundancy has been reduced but strong correlations may still remain among features. To further compress the dimension and organise the input structure, we apply PCA and ICA in sequence at this stage.

PCA compresses highly correlated feature variation into a smaller number of principal directions. This reduces the input dimension and at the same time alleviates linear redundancy across features. After PCA, the original features are transformed into a set of principal components that retain the main variance information of the data.

We then apply ICA after PCA to separate latent signal sources that remain mixed. Let H_t denote the PCA output. ICA expresses H_t as a linear mixture of latent components:

$$H_t = A\,S_t,$$

where A is the mixing matrix and S_t represents the latent independent components. The goal of ICA is to estimate a separation matrix W such that

$$\hat{S}_t = W\,H_t.$$

PCA mainly handles dimensionality reduction and linear correlation, while ICA further separates mixed variation signals on top of PCA. After the two-stage processing, the input becomes more compact and more convenient for downstream forecasting models. The numbers of principal components retained by PCA and of independent components extracted by ICA are specified in Section 4.

3.3.4 Output of Reconstructed Features

After feature filtering, local normalisation, and the two-stage PCA–ICA reconstruction, the raw observation window X_t^{raw} is mapped to the reconstructed input matrix $Z_t \in \mathbb{R}^{L_x \times d_{\text{in}}}$, where d_{in} is the feature dimension after reconstruction. Figure 3 summarises the dimension changes across the preprocessing pipeline. Z_t serves as the actual input to the downstream Transformer encoder, which produces predictions for the next L_y time steps.

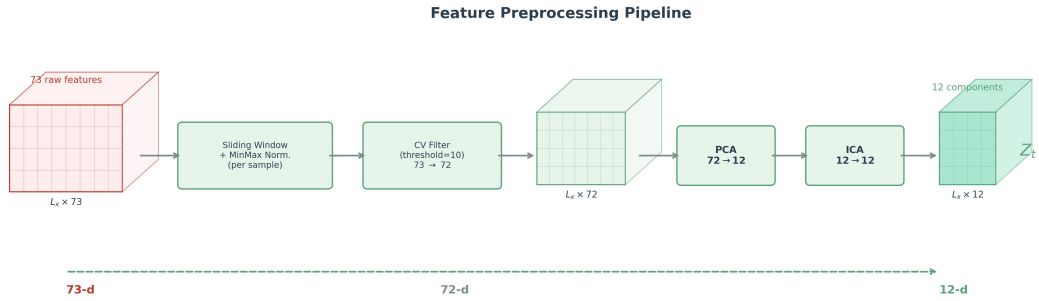

Figure 3. Dimension Changes in Feature Reconstruction. The raw input $L_x \times 73$ passes through sample-level Min-Max normalisation, CV filtering ($73 \to 72$), PCA ($72 \to 12$), and ICA ($12 \to 12$), yielding $Z_t \in \mathbb{R}^{L_x \times 12}$.

3.4 Transformer-based Direct Multi-step Forecasting Architecture

With the reconstructed input Z_t, we use a Transformer encoder–decoder to learn the mapping between the observation window and the forecasting horizon. The model consists of three parts: the encoder, the decoder, and the decoder input design. Both the encoder and the decoder use a 6-layer stacked structure. The model outputs the full forecasting window in a single forward pass, without step-by-step recursive extrapolation. In our implementation, the encoder operates directly on the reconstructed feature dimension d_{in}; that is, $d_{\text{model}} = d_{\text{in}}$, and no additional embedding layer is introduced.

3.4.1 Encoder

The encoder maps the reconstructed feature sequence over the observation window to an encoded representation. Let

$$Z_t = \{z_{t-L_x+1}, \ldots, z_t\} \in \mathbb{R}^{L_x \times d_{\text{in}}},$$

where $z_\tau \in \mathbb{R}^{d_{\text{in}}}$ denotes the reconstructed feature vector at time τ. Since the self-attention mechanism itself does not contain temporal order information, positional encoding is added to the input before it enters the encoder so that the order of each time position is preserved. The encoder extracts dependencies within the observation window layer by layer through multi-head self-attention and feed-forward networks, producing the encoded representation $H_t = \text{Encoder}(Z_t) \in \mathbb{R}^{L_x \times d_{\text{in}}}$. H_t preserves the temporal information from the observation window and serves as the basis for the decoder to generate predictions.

3.4.2 Decoder: Direct Multi-step Forecasting

After obtaining the encoded representation H_t, the model generates predictions for the next L_y time steps through the decoder. The decoder input D_t (the specific construction is given in Section 3.4.3) is a scalar sequence of length $L_y \times 1$. It first passes through a linear projection layer that expands it to $L_y \times d_{\text{in}}$ so that its dimension matches the encoder output, and is then combined with positional encoding.

Within each decoder layer, data pass through three sub-modules in sequence: masked multi-head self-attention (modelling temporal dependencies within the forecasting window under a causal mask), multi-head cross-attention using H_t as keys and values (passing observation-window information into the decoder), and a feed-forward network. Each sub-module is followed by a residual connection and layer normalisation.

After six decoder layers, the output is compressed from d_{in} to 1 through a linear projection, giving the final predicted sequence

$$\hat{Y}_t = \text{Linear}\big(\text{Decoder}(D_t, H_t)\big) \in \mathbb{R}^{L_y}.$$

Unlike recursive forecasting, the prediction at one step is not fed back into the model; instead, a forecasting sequence of length L_y is produced directly in a single forward pass, preventing early-step errors from propagating at inference.

3.4.3 Decoder Input Design

To ensure that the start of the forecasting window connects naturally with the end of the observation window, we design the decoder input specifically for this task. Let the single-step return at the end of the observation window be denoted by

$$g_t = \frac{p_t}{p_{t-1}} - 1.$$

In the training mode, the decoder input is constructed by concatenating this observed starting item with the subsequent ground-truth target sequence:

$$D_t^{\text{train}} = [g_t, r_{t+1|t}, r_{t+2|t}, \ldots, r_{t+L_y-1|t}].$$

A causal mask is applied during training so that the decoder can only attend to earlier positions. In this way, the model learns to transition from the end of the observation window to the forecasting window.

At inference, future values are not available, and g_t is repeated L_y times to serve as the decoder input:

$$D_t^{\text{infer}} = [g_t, g_t, \ldots, g_t].$$

Compared with the zero-padding used in Informer, g_t carries information about the most recent price change at the end of the observation window, so that the start of the forecasting window is no longer a blank signal. D_t is then expanded to d_{in} dimensions through a linear projection and combined with positional encoding before entering the decoder (see Figure 4).

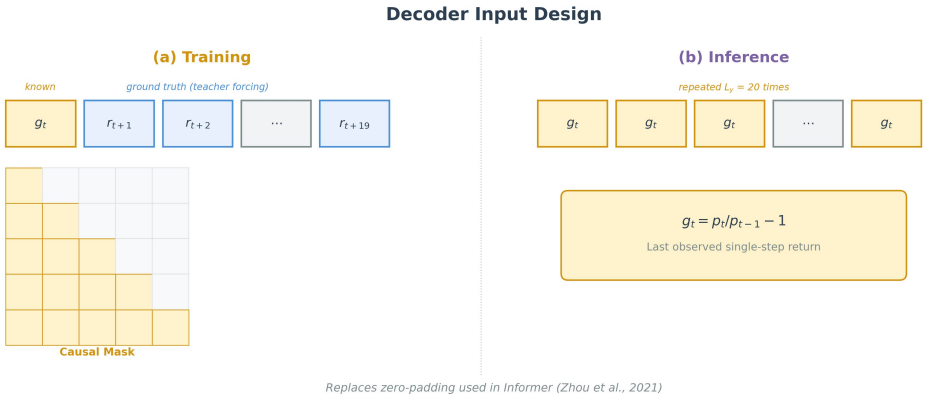

Figure 4. Decoder Input Design. (a) Training stage: the starting item is g_t, followed by $L_y - 1$ ground-truth targets, under a causal mask. (b) Inference stage: g_t is repeated L_y times. This design replaces the zero-padding used in Informer (Zhou et al., 2021).

3.4.4 Output Form and Model Training

The model outputs relative return predictions for L_y time steps. At the evaluation stage, the predicted relative returns are converted to closing prices through the inverse transformation $\hat{p}_{t+j} = (\hat{r}_{t+j|t} + 1)\, p_t$. Model performance can therefore be evaluated on both the reconstructed closing-price sequence and the relative-return sequence output by the model. Taking mean squared error (MSE) as an example, the training loss is written as

$$L(\theta) = \frac{1}{L_y} \sum_{j=1}^{L_y} (\hat{r}_{t+j|t} - r_{t+j|t})^2,$$

where θ denotes the parameters of the Transformer forecasting model. Only the parameters of the forecasting model itself are updated through backpropagation during training. The earlier steps of feature filtering, local normalisation, and PCA–ICA processing are completed before the sample enters the model and are not jointly updated through gradients during training. Training details such as the choice of optimiser, learning rate schedule, and early stopping are specified in Section 4.

4 Experimental Setup

This section specifies the concrete configuration used to implement the method of Section 3, covering in order the dataset, sample construction, feature reconstruction, compared models, training configuration, the L_x experimental design, and the evaluation metrics.

4.1 Dataset

We use the constituents of the SSE 180 Index in the Chinese A-share market as the stock pool. This index is chosen for two reasons. First, it covers the 180 large-cap, liquid stocks listed on the Shanghai Stock Exchange, spanning finance, manufacturing, resources, technology, and other major sectors, with a relatively balanced industry distribution. Second, the data anomalies among its constituents, such as trading suspensions and ex-rights adjustments, are limited, making it convenient to construct historical

series of uniform length.

The target stock is Luoyang Molybdenum (stock code 603993), a non-ferrous metal smelting and processing firm. The stock was randomly drawn from the index constituents rather than pre-selected, avoiding the bias of ex-post cherry-picking. For each stock we collect 1,024 trading days (about four years) of daily data, including open, high, low, close, volume, and commonly used technical indicators, totalling 73 raw features. Table 1 summarises the descriptive statistics of the target stock.

Table 1. Descriptive Statistics of the Target Stock 603993.

Item	Value
Sample period	2021-12-03 to 2026-02-27
Number of trading days	1,024
Industry	Non-ferrous metals (smelting and processing)
Closing price, mean / median	7.72 / 6.14 CNY
Closing price, standard deviation	4.15 CNY
Closing price, range	[4.15, 27.03] CNY
Cumulative return over the period	$+293.9\%$ (6.09 $\rightarrow$ 23.99 CNY)
Daily log return, mean	0.134%
Daily log return, standard deviation	2.593%
Annualised volatility ($\times\sqrt{252}$)	41.17%
Daily log return, skewness	-0.03
Daily log return, excess kurtosis	1.94
Daily log return, first-order autocorrelation	0.024
Average daily turnover	0.99%
Jarque–Bera normality test	Normality rejected ($p < 0.001$)

Note. The daily log return is computed as the log difference of closing prices on adjacent trading days. The annualised volatility is scaled by 252 trading days.

The descriptive statistics show that the stock experienced substantial price appreciation over the sample period (cumulative $+293.9\%$), with an annualised volatility of about 41%. The daily log return series is approximately symmetric (skewness -0.03) but exhibits mildly heavier tails than a normal distribution (excess kurtosis 1.94); the Jarque–Bera test correspondingly rejects normality, though the magnitude of departure is modest. The first-order autocorrelation is close to zero, indicating that the linear predictability of the daily return series is weak. We analyse 603993 as a single case rather than a "representative sample": using a single target stock is intended to fix the object of study and focus on comparing the effects of different observation window lengths and model architectures. The generalisability of single-stock conclusions is discussed in Section 6.5.

At the similar-stock search stage, we use a fixed 250-trading-day (approximately one-year) closing price series as the search window, independent of the observation window length L_x used by the downstream forecasting model. The choice of 250 days is intended to capture medium-to-long-term price shape similarity. This search window is fixed throughout the search stage and does not vary with L_x. The 250-day price series used for search lies entirely within the training-set period and does not use any price information from the validation or test periods; thus, there is no look-ahead bias.

The details of the similarity measures and the voting mechanism have been given in Section 3.2.2. Under the experimental configuration of this paper, the similar-stock set for Luoyang Molybdenum includes four stocks: Zijin Mining (601899), GigaDevice Semiconductor (603986), Jiangxi Copper (600362), and TBEA (600089). The target stock and the four similar stocks—five stocks in total—with

all 73 raw features jointly form the augmented sample pool. Since the five stocks are drawn from the same period and have close price shapes, the merged samples exhibit strong cross-sectional correlation; Section 6 will take this into account when interpreting the model comparisons.

Table 2. Dataset Summary.

Item	Value
Stock pool	SSE 180
Target stock	603993 (Luoyang Molybdenum)
Number of similar stocks M	4 (601899, 603986, 600362, 600089)
Similar-stock search window	250 trading days (fixed)
Voting threshold	$4/6$
Number of trading days per stock	1,024
Number of raw features	73
Number of features after CV filtering	72
Input dimension after two-stage PCA–ICA (d_{in})	12
Forecasting horizon L_y	20 (fixed)
Training / validation / test split	80%/10%/10% (chronological)

4.2 Sequence Construction and Data Split

On top of the augmented sample pool, we generate training samples with a sliding window. Let the observation window length be L_x and the forecasting horizon $L_y = 20$ trading days. For each stock's 1,024-day data, the window slides from start to end with stride one: at each step, a historical window of length L_x is extracted as the input sequence, and the subsequent L_y time steps serve as the prediction target. Taking $L_x = 120$ as an example, a single stock of 1,024 days yields $1{,}024 - 120 - 20 + 1 = 885$ samples. Some of the similar stocks have slightly fewer than 1,024 available trading days due to suspensions or listing dates; the five stocks therefore yield approximately 4,000 sliding-window samples in total.

The split proceeds in two steps. First, the test set is fixed chronologically as the most recent contiguous 10% of the target stock's samples; reserving a continuous tail segment for testing prevents information leakage from the target stock's future into the training process, which would otherwise arise because adjacent sliding windows overlap heavily (stride one). Second, the remaining samples—the target stock's earlier 90% together with all sliding-window samples of the four similar stocks—are pooled and randomly shuffled, and 10% of the target stock's share is drawn to form the validation set; the rest becomes the training set. Taking $L_x = 120$ as an example, the resulting split contains approximately 3,896 training, 89 validation, and 88 test samples. The deep learning models therefore use substantially more training samples than ARMAGARCH, which is fitted independently on each test window of the target stock alone (see Section 4.4).

4.3 Feature Reconstruction

Following the feature reconstruction framework in Section 3.3, the 73 raw features pass through three processing steps before entering the model.

Step 1: CV filtering. The threshold is set to 10. In our experiment, one low-variation feature is filtered out ($73 \to 72$ dimensions).

Step 2: Sample-level two-step normalisation. Min-Max scaling and zero-mean centring are performed independently within each window, without cross-sample information leakage.

Step 3: Two-stage PCA–ICA dimensionality reduction. PCA retains 12 principal components, and ICA further extracts 12 independent components. The choice of 12 dimensions is intended to control the overfitting risk of deep learning models under the approximately 4,400-sample scale. The ablation in Section 5.4.2 directly examines this choice. The processed input dimension is $73 \rightarrow 12$.

The three preprocessing steps strictly follow the training–validation–test split. The CV filtering rules and the projection matrices of PCA and ICA are fitted on the training set only and then applied to the validation and test sets. The sample-level normalisation, by construction, does not involve cross-split fitting.

4.4 Compared Models

We set up six model configurations for experimental comparison. The six configurations differ in architecture, parameter count, and internal dimension, and the comparison reflects each model's predictive performance under its default configuration. The core structures of the models are summarised in Table 3.

(1) TransED — encoder–decoder Transformer, nhead $= 3$**.** The encoder–decoder Transformer proposed in this paper. The encoder and the decoder each have six layers, with three attention heads, a feed-forward dimension of 2,048, and dropout 0.1. The input dimension is $d_{\text{in}} = 12$, and the PCA–ICA output dimension is used directly as d_{model} without an additional embedding layer. The decoder input follows the g_t initialisation scheme of this paper (see Section 3.4.3). The parameter count is approximately 633K.

(2) TransED — nhead $= 6$**.** Only the number of attention heads in (1) is changed from three to six. The total parameter count of multi-head attention is independent of the head count, so the parameter count remains 633K. This configuration is used to examine the effect of the number of attention heads on predictive performance.

(3) TransE — encoder-only Transformer. Only the Transformer encoder is used, without a decoder. The input is linearly projected from 12 to 128 dimensions and then fed to the encoder. The encoder has six layers, eight attention heads, and dropout 0.1. The encoder output passes through two linear projections to produce the L_y-step prediction, with the two layers handling dimension compression and time-step mapping, respectively. The parameter count is approximately 4,155K, about 6.5 times that of TransED. The difference in d_{model} between TransE and TransED (128 vs. 12) reflects the natural choice of each architecture. The ablation in Section 5.4.2 directly examines whether enlarging d_{model} benefits TransED.

(4) BiLSTM — bidirectional LSTM. A classical deep learning baseline. A 2-layer bidirectional LSTM with a hidden dimension of 128 and dropout 0.1. The LSTM output passes through two linear projections to produce the L_y-step prediction. The parameter count is approximately 543K, comparable to that of TransED.

(5) ARMAGARCH — autoregressive moving average with GARCH. A classical statistical baseline. After first-differencing the closing prices of the target stock and applying Min-Max normalisation, ARMA and GARCH models are fitted separately. The lag orders p and q of ARMA are selected automatically within the range 1 to 10 through a grid search based on the Akaike Information Criterion. ARMAGARCH

re-estimates its parameters independently on each evaluation window, using only the information available within that window. This is fundamentally different from the "train once, infer on the whole test set" procedure of the deep learning models. This model runs on CPU.

(6) TransED-Embed — TransED with an embedding layer, $d_{\text{model}} = 128$, nhead $= 8$. A linear embedding layer is added at the input of TransED to project the input from 12 to 128 dimensions, so that the Transformer computes attention in a 128-dimensional space, with eight attention heads of dimension 16 each. The parameter count is approximately 7,523K, about 12 times that of the original TransED. This configuration is used to examine whether enlarging the internal dimension improves the predictive performance of TransED.

Table 3. Comparison of the Six Model Configurations.

Model	Structure	d_{model}	Layers	Attention heads	Parameters
TransED (nh3)	Encoder–decoder	12	$6+6$	3	633K
TransED (nh6)	Encoder–decoder	12	$6+6$	6	633K
TransE	Encoder-only	128	6	8	4,155K
BiLSTM	Bidirectional LSTM	128	2	–	543K
ARMAGARCH	Statistical model	1	–	–	–
TransED-Embed	Encoder–decoder + embedding	128	$6+6$	8	7,523K

4.5 Training Configuration

All deep learning models except ARMAGARCH share a common training strategy to ensure comparability. The optimiser is AdamW with a learning rate of 1×10^{-5} and weight decay 0.01. The learning rate follows a cosine annealing schedule with a minimum of 1×10^{-6}. The batch size is 128, and the maximum number of training epochs is 300. The loss function is MSE.

The prediction target is the relative return $r_{t+j|t} = p_{t+j}/p_t - 1$. At the evaluation stage, the inverse transformation $\hat{p}_{t+j} = (\hat{r}_{t+j|t} + 1)\, p_t$ recovers the price prediction.

Early stopping is used during training: if the validation loss does not reach a new minimum for 10 consecutive epochs, training is terminated early, and the model parameters corresponding to the lowest validation loss are loaded for subsequent evaluation.

All deep learning experiments run under PyTorch 2.11.0, using an Apple Silicon GPU (Metal Performance Shaders backend) for acceleration. ARMAGARCH runs on CPU. The random seed is set to 42 for all experiments.

4.6 Systematic Experiment on Observation Window Length

The central experimental design of this paper is to systematically vary the observation window length L_x, to examine whether different models respond differently to the choice of window length. For each model configuration, $L_y = 20$ and all other training hyperparameters are held fixed, and L_x takes the ten values $\{120, 140, 160, 180, 200, 220, 240, 260, 280, 300\}$ in turn, with each L_x trained and evaluated independently. The stride of 20 days is sufficient to capture possible non-monotonic variation of RMSE with L_x while keeping the computational cost of a single run manageable.

TransED (nhead = 3 and nhead = 6), TransE, and ARMAGARCH are evaluated at all ten values of L_x. BiLSTM serves as a reference deep learning baseline and is evaluated at eight representative values $L_x \in \{120, 140, \ldots, 260\}$, sufficient to characterise its dependence on window length. TransED-Embed is evaluated at three representative window lengths, $L_x \in \{120, 180, 240\}$, covering both ends and the middle of the U-shaped curve of the TransED series while keeping the running time of the ablation experiment manageable.

4.7 Evaluation Metrics

Models are evaluated on the test set using the metrics listed in Table 4. The predicted output is the relative return; for metrics defined on closing prices, the inverse transformation is applied first. All metrics are computed per sample across the L_y prediction steps and then averaged over the test set.

RMSE, mean absolute error (MAE), and mean absolute percentage error (MAPE) are common point forecast metrics. Directional accuracy and trend accuracy measure the model's ability to judge the direction of price changes at two scales: per-step direction and overall segment direction, respectively. The correlation coefficient reflects the linear consistency between the predicted series and the true series.

Table 4. Evaluation Metrics.

Category	Metric	Formula	Direction
Point accuracy	RMSE	$\sqrt{\text{mean}((y-\hat{y})^2)}$	smaller
Point accuracy	MAE	$\text{mean}(\lvert y-\hat{y}\rvert)$	smaller
Point accuracy	MAPE	$\text{mean}(\lvert y-\hat{y}\rvert/\lvert y\rvert)$	smaller
Direction	Directional accuracy	Fraction of steps with matching direction	larger
Direction	Trend accuracy	Fraction of L_y-step net direction matches	larger
Direction	Correlation	Pearson correlation coefficient	larger
Baseline	R^2_{hist}	$1-\sum(y-\hat{y})^2/\sum(y-\bar{y}_{\text{hist}})^2$	larger
Baseline	$\text{MASE}_{\text{naive}}$	$\text{mean}(\lvert y-\hat{y}\rvert)/\text{mean}(\lvert y-y_{\text{hist}[-1]}\rvert)$	smaller (<1 beats)

For non-stationary stock price series, standard R^2 and standard MASE provide limited interpretive value. Standard R^2 uses the mean of the test targets as the baseline, which is not particularly informative in a non-stationary setting. Standard MASE compares the model's error against the error of a seasonal-naive forecast, which is designed for series with known periodicity (e.g., monthly or weekly cycles). Daily stock prices do not have a natural seasonality of this kind, so this benchmark does not correspond to a meaningful comparator in the present setting. We therefore use two explicitly defined baseline-referenced metrics. R^2_{hist} takes the mean closing price within the observation window as the baseline and measures the model's improvement over the historical-mean predictor. $\text{MASE}_{\text{naive}}$ takes the last closing price in the observation window as the baseline and measures the model's improvement over a persistence (random-walk) predictor. Under an approximately random-walk price process, $\text{MASE}_{\text{naive}}$ is usually the more stringent benchmark. These two metrics are explicitly defined for this paper and are not equivalent to standard R^2 or standard MASE variants in the literature that use different baselines.

5 Experimental Results

This section reports the predictive performance of the six models under different observation window lengths. All numerical values are averaged over the full test set. We use RMSE as the primary point forecast metric, with directional accuracy, $\text{MASE}_{\text{naive}}$, and R^2_{hist} as complementary metrics. The remaining metrics—MAE, MAPE, trend accuracy, and correlation coefficient—serve as consistency cross-checks.

The main empirical finding is direct: under our experimental setting, no deep learning model consistently outperforms the classical ARMAGARCH baseline across the tested configurations. Specifically, ARMAGARCH achieves the lowest RMSE across all ten observation windows, outperforming the best deep learning model TransE by approximately 1.4% to 17.0%. TransE itself exhibits a non-monotonic first-decrease-then-increase pattern: its RMSE drops by 8.1% over the interval $L_x \in [120, 240]$, reaches the minimum of 1.687 at $L_x = 240$, and rebounds by about 3.2% at $L_x = 300$. Enlarging the internal dimension of TransED from 12 to 128 worsens the average RMSE by 11.6% and pushes the correlation coefficient from 0.398 down to -0.069. In contrast, increasing the number of attention heads from three to six affects the average RMSE by only about 0.7%. The following subsections examine these observations in detail.

5.1 Overall Results

The RMSE, MAE, MAPE, and directional accuracy of all models under different observation window lengths are summarised in Table 5 through Table 8.

Table 5. Panel A — RMSE (Lower Is Better; Bold = Global Optimum per Row; Italic = Deep-Learning Optimum per Row).

L_x	TransED (nh3)	TransED (nh6)	TransE	BiLSTM	ARMAGARCH	TransED-Embed
120	1.858	1.894	*1.835*	2.125	**1.569**	2.114
140	1.876	1.848	*1.842*	2.071	**1.641**	–
160	1.901	1.916	*1.834*	1.986	**1.668**	–
180	1.835	1.840	*1.796*	1.908	**1.679**	2.057
200	1.936	1.958	*1.796*	1.896	**1.654**	–
220	1.851	1.774	*1.701*	1.802	**1.641**	–
240	1.807	1.842	*1.687*	2.032	**1.633**	1.970
260	1.835	1.812	*1.695*	1.731	**1.672**	–
280	1.804	1.860	*1.714*	–	**1.689**	–
300	1.767	1.859	*1.741*	–	**1.689**	–

Note. "–" indicates that the model was not evaluated at the given L_x. TransED-Embed is evaluated only at three representative windows ($L_x = 120, 180, 240$), and BiLSTM is evaluated up to $L_x = 260$ as a reference deep learning baseline. The same convention applies to Panels B, C, and D below.

Overall, ARMAGARCH achieves the lowest RMSE across all windows and the best directional accuracy in seven out of eight common test windows; the MAE and MAPE rankings are fully consistent with the RMSE ranking (detailed in Section 5.2). Among the deep learning models, TransE and TransED are closest to ARMAGARCH, while BiLSTM and TransED-Embed are systematically weaker.

Three observations can be drawn from Figure 5. First, ARMAGARCH consistently lies in the top-performing region in both RMSE and directional accuracy, with a nearly flat curve, reflecting its insensitivity to L_x. Second, the RMSE curve of TransE exhibits a clear first-decrease-then-increase

Table 6. Panel B — Directional Accuracy (Higher Is Better; %).

L_x	TransED (nh3)	TransED (nh6)	TransE	BiLSTM	ARMAGARCH	TransED-Embed
120	55.02	54.67	52.87	50.42	**58.43**	49.58
140	51.65	55.81	54.83	52.33	**56.06**	–
160	54.32	54.89	52.94	52.69	**55.70**	–
180	54.11	**56.16**	50.77	50.96	55.07	49.29
200	53.36	53.75	48.68	51.25	**55.33**	–
220	53.58	54.39	55.33	51.75	**55.87**	–
240	52.98	52.91	53.05	49.31	**55.68**	50.14
260	53.77	53.56	50.92	50.71	**55.05**	–
280	**54.46**	53.44	52.19	–	53.65	–
300	**54.81**	53.91	52.93	–	53.65	–

Table 7. Panel C — MAE (Lower Is Better).

L_x	TransED (nh3)	TransED (nh6)	TransE	BiLSTM	ARMAGARCH	TransED-Embed
120	1.578	1.619	*1.559*	1.764	**1.328**	1.827
140	1.609	1.589	*1.568*	1.750	**1.393**	–
160	1.627	1.656	*1.563*	1.702	**1.425**	–
180	1.580	1.588	*1.532*	1.637	**1.437**	1.779
200	1.674	1.692	*1.529*	1.610	**1.412**	–
220	1.582	1.516	*1.440*	1.525	**1.397**	–
240	1.543	1.572	*1.420*	1.686	**1.383**	1.684
260	1.569	1.549	*1.429*	1.452	**1.416**	–
280	1.540	1.594	*1.443*	–	**1.429**	–
300	1.497	1.593	*1.466*	–	**1.429**	–

trajectory, with the minimum located near $L_x \approx 240$; in contrast to ARMAGARCH, both TransE and TransED reach their best RMSE at longer observation windows, suggesting a stronger dependence on the length of the input sequence. Third, BiLSTM and TransED-Embed lie noticeably below the main cluster in both metrics, confirming their overall weaker performance. The heatmap in Figure 6 reinforces this distribution through colour. The row corresponding to ARMAGARCH is visibly lighter than the others, while the row for TransED-Embed is the darkest, peaking near $L_x = 180$.

5.2 Point Forecast Accuracy

ARMAGARCH achieves the lowest RMSE across all ten tested windows. Its RMSE fluctuates between 1.569 and 1.689, with a total variation of only 7.7%, making it the least window-sensitive model. A mild upward drift in RMSE can be observed as L_x increases, with the lowest value reached at $L_x = 120$ and the highest at $L_x = 300$. At $L_x = 120$, its RMSE is already 14.5% below that of TransE, the best-performing deep learning model.

TransE is the deep learning configuration with the lowest RMSE, and its curve follows a non-monotonic first-decrease-then-increase pattern as L_x varies. The RMSE is 1.835 at $L_x = 120$, reaches the minimum of 1.687 at $L_x = 240$ (an improvement of about 8.1% over the interval), and then rebounds over $L_x \in [260, 300]$. Nevertheless, TransE never reaches the RMSE level of ARMAGARCH within the entire tested range of L_x.

The RMSE of TransED exhibits a U-shaped curve as L_x varies. The short window ($L_x = 120$, RMSE

Table 8. Panel D — MAPE (Lower Is Better; %).

L_x	TransED (nh3)	TransED (nh6)	TransE	BiLSTM	ARMAGARCH	TransED-Embed
120	8.29	8.48	*8.17*	9.31	**7.13**	9.63
140	8.40	8.30	*8.17*	9.11	**7.38**	–
160	8.40	8.58	*8.10*	8.82	**7.47**	–
180	8.13	8.15	*7.88*	8.41	**7.47**	9.19
200	8.53	8.61	*7.78*	8.20	**7.26**	–
220	7.95	7.65	*7.26*	7.69	**7.08**	–
240	7.70	7.86	*7.10*	8.42	**6.93**	8.46
260	7.82	7.73	*7.12*	7.25	**7.07**	–
280	7.65	7.91	*7.15*	–	**7.08**	–
300	7.38	7.85	*7.23*	–	**7.08**	–

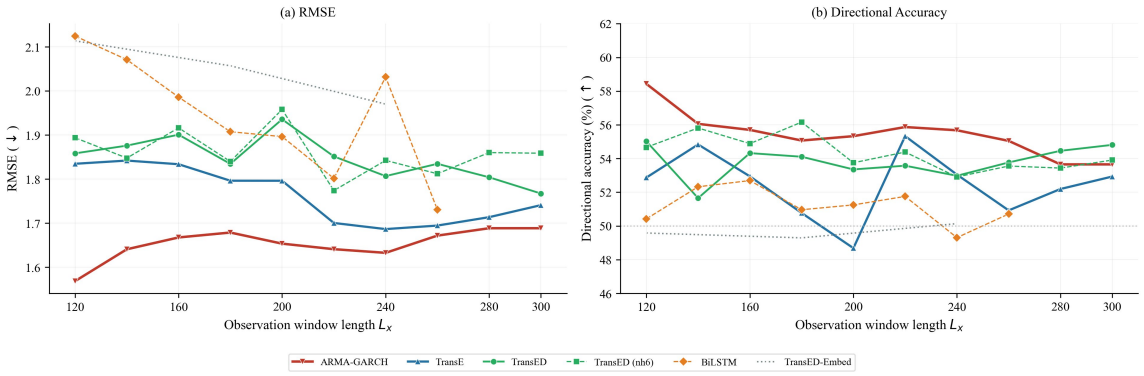

Figure 5. RMSE (Left) and Directional Accuracy (Right) of the Six Model Configurations as a Function of Observation Window Length L_x.

$= 1.858$) and the long window ($L_x = 300$, RMSE $= 1.767$) both perform relatively well, while the middle region near $L_x \approx 200$ attains the peak. The nhead $= 6$ configuration shows the same shape. This "worst-in-the-middle" pattern is the opposite of the "first-decrease-then-increase" pattern of TransE.

The RMSE of BiLSTM fluctuates across windows by up to 22.8%, the largest among all models, and shows no monotonic trend. This instability is consistent with the vanishing or exploding gradients of recurrent structures on long sequences and their sensitivity to initialisation.

TransED-Embed yields higher RMSE than the original TransED at all three tested windows, with an average deterioration of 11.6% (see Section 5.4.2).

The rankings by MAE and MAPE are fully consistent with those by RMSE. ARMAGARCH attains the global minimum at all ten windows, TransE attains the best deep learning result at each window, and the optimal L_x locations of all three metrics coincide with one another. Taken together, the advantage of ARMAGARCH over the deep learning models is robust.

5.3 Directional Prediction and Comparison with Baselines

5.3.1 Directional Accuracy

Across the eight common test windows ($L_x = 120$–260), ARMAGARCH achieves the highest directional accuracy in seven windows, with a maximum of 58.43% at $L_x = 120$. TransED (nhead $= 6$) reaches

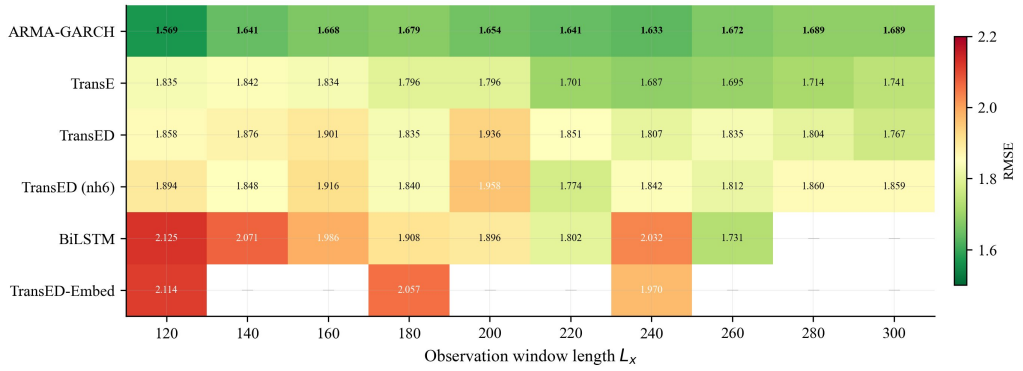

Figure 6. RMSE Heatmap. Darker colours indicate larger errors; "–" marks windows in which no experiment was conducted.

Table 9. TransE RMSE at Selected L_x and Gap Relative to ARMAGARCH.

L_x	TransE RMSE	Gap vs. ARMAGARCH
120	1.835	+17.0%
200	1.796	+8.6%
240	1.687	+3.3%
260	1.695	+1.4%
300	1.741	+3.1%

56.16% at $L_x = 180$, and is the only deep learning configuration to surpass ARMAGARCH in a common window. Over the longer windows not covered by BiLSTM ($L_x = 280$ and 300), TransED (nh3) is slightly higher than ARMAGARCH, but the margin is small. Overall, the directional advantage of ARMAGARCH is stable but does not cover every window.

The directional accuracy of BiLSTM stays within 49–53% across all windows, with most values close to 50%. TransED-Embed yields 49.58%, 49.29%, and 50.14% at the three tested windows, all at or below 50%. A directional accuracy near 50% is effectively indistinguishable from random guessing, and these models do not provide stable directional signals under the current setting.

5.3.2 $\text{MASE}_{\text{naive}}$ *and Correlation Coefficient*

Only two models keep $\text{MASE}_{\text{naive}}$ below one across all tested windows: ARMAGARCH ($\text{MASE}_{\text{naive}} \in [0.944, 0.973]$) and TransE ($\text{MASE}_{\text{naive}} \in [0.953, 0.973]$). TransED (nh3) shows $\text{MASE}_{\text{naive}}$ fluctuating within $[0.967, 1.068]$, with roughly half of the windows below one and the other half above one. BiLSTM exceeds 1.0 in most windows, reaching a maximum of 1.160. TransED-Embed is above one at all three tested windows, within $[1.14, 1.18]$.

Most deep learning models fail to beat the naive baseline. A simple strategy that repeats the closing price at the end of the observation window throughout the full $L_y = 20$ prediction steps is already stronger than most deep learning configurations, including BiLSTM and TransED-Embed. For a target that behaves largely as a random walk, the naive baseline itself is already a strong benchmark.

In contrast, the other baseline-referenced metric, R^2_{hist}, is positive for all 51 model–window combinations evaluated and lies within a narrow range of 0.91 to 0.97 across models. Even the lowest R^2_{hist}, obtained by TransED-Embed at $L_x = 120$ (0.907), is far from zero. Over a 20-step forecasting horizon, the

Table 10. TransED vs. TransED-Embed RMSE at Three L_x.

L_x	TransED (nh3) RMSE	TransED-Embed RMSE	Deterioration
120	1.858	2.114	+13.7%
180	1.835	2.057	+12.1%
240	1.807	1.970	+9.0%

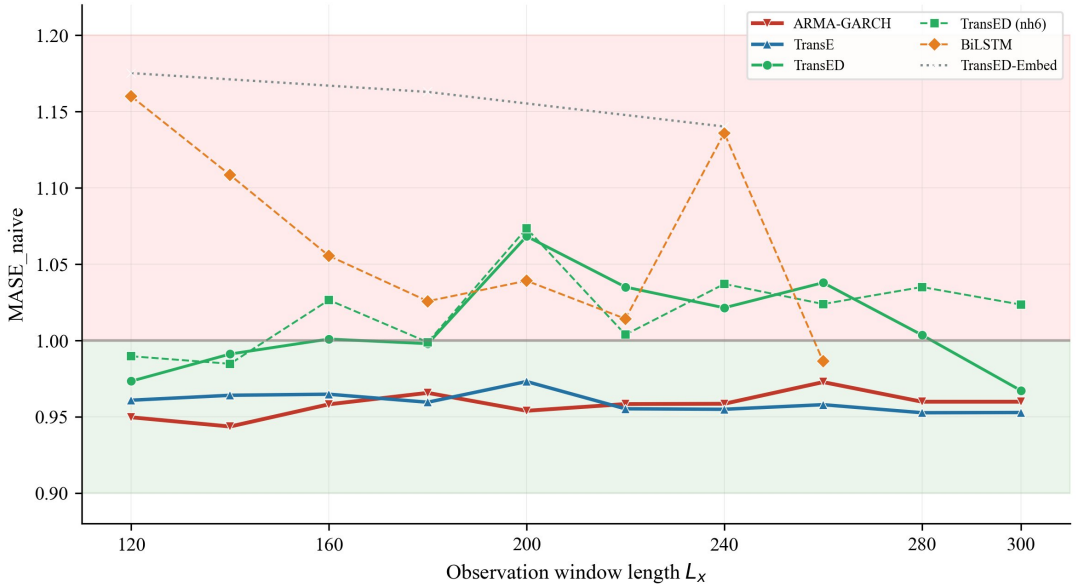

Figure 7. $\text{MASE}_{\text{naive}}$ of Each Model as a Function of L_x. The green region corresponds to $\text{MASE}_{\text{naive}} < 1$ (better than the naive baseline), and the red region to $\text{MASE}_{\text{naive}} > 1$ (worse than the naive baseline).

mean closing price within the observation window is a weak benchmark, because the true prices typically drift away from the window mean; any model that tracks the general price level at all will easily exceed it. This contrast between R^2_{hist} and $\text{MASE}_{\text{naive}}$ directly supports the design choice in Section 4.7: R^2_{hist} is retained for transparency as a classical baseline reference, but $\text{MASE}_{\text{naive}}$ provides the more demanding benchmark and therefore serves as the primary baseline-referenced metric in this study.

For the correlation coefficient, ARMAGARCH and the TransED series stay within 0.34–0.42. TransE stays within 0.25–0.39. BiLSTM is close to zero with large fluctuation. TransED-Embed is negative at most windows, and recovers to only 0.056 at $L_x = 240$. A negative correlation means that the predicted series tends to move in the opposite direction to the true series, which is typically consistent with overfitting or a mismatch between the learned representation and the target.

5.4 Ablation Studies

5.4.1 Number of Attention Heads (nhead = 3 *vs.* 6)

Under the $d_{\text{model}} = 12$ constraint, we increase the number of attention heads from three to six while keeping all other configurations unchanged.

In RMSE, nhead = 3 is better in 7 out of 10 windows and nhead = 6 in the remaining three, with an average RMSE difference of 0.7%. The average directional accuracy differs by only 0.54 percentage points. A closer look at directional accuracy shows that nhead = 6 tends to be better at middle windows

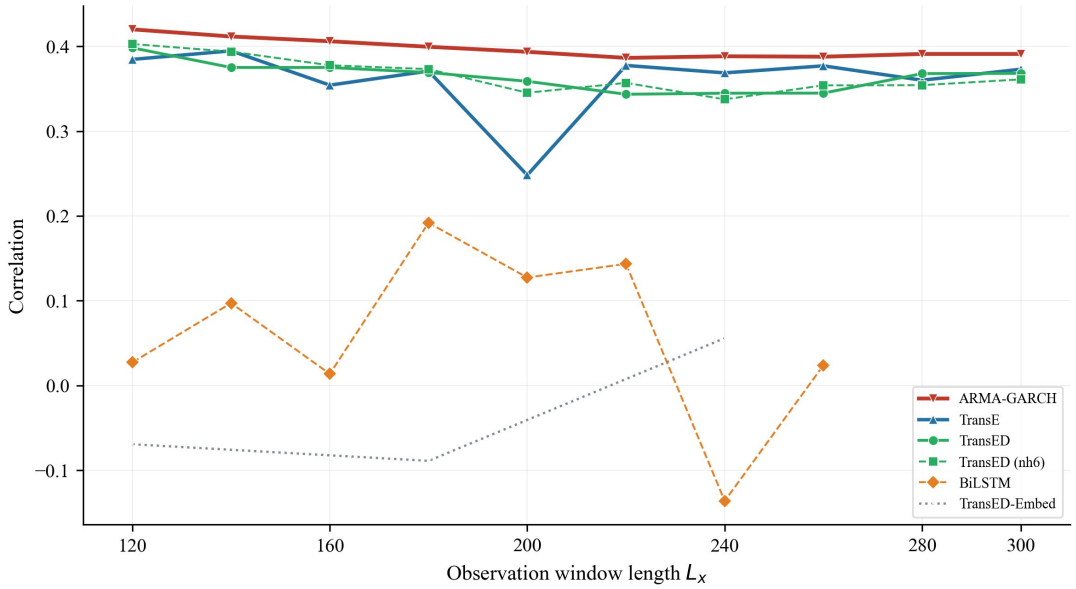

Figure 8. Correlation Coefficient of Each Model as a Function of L_x. Pearson correlation between the predicted and true series.

Table 11. TransED: nhead = 3 vs. nhead = 6.

L_x	nh3 RMSE	nh6 RMSE	Diff.	nh3 dir. (%)	nh6 dir. (%)	Diff. (pp)
120	1.858	1.894	+1.9%	55.02	54.67	−0.35
140	1.876	1.848	−1.5%	51.65	55.81	+4.16
160	1.901	1.916	+0.8%	54.32	54.89	+0.57
180	1.835	1.840	+0.3%	54.11	56.16	+2.05
200	1.936	1.958	+1.1%	53.36	53.75	+0.39
220	1.851	1.774	−4.2%	53.58	54.39	+0.81
240	1.807	1.842	+2.0%	52.98	52.91	−0.07
260	1.835	1.812	−1.2%	53.77	53.56	−0.21
280	1.804	1.860	+3.1%	54.46	53.44	−1.02
300	1.767	1.859	+5.2%	54.81	53.91	−0.90
Mean	1.847	1.860	+0.7%	53.81	54.35	+0.54

($L_x = 140$ to 220, five consecutive windows), while nhead = 3 tends to be better at the two ends ($L_x = 120$, and $L_x = 240$ to 300). Although neither configuration dominates the other on average, nhead = 6 attains the single best RMSE in this ablation (1.774 at $L_x = 220$) and the highest directional accuracy (56.16% at $L_x = 180$), indicating a slightly higher upper bound under favourable window lengths. Under the $d_{\text{model}} = 12$ constraint, each head has a dimension of four under nhead = 3 and only two under nhead = 6. Both values are low, and the effect of the head count on predictive performance is therefore limited.

5.4.2 Internal Model Dimension (TransED vs. TransED-Embed)

TransED-Embed adds a linear embedding layer at the input of the original TransED, projecting the input from 12 to 128 dimensions so that the Transformer computes attention in a 128-dimensional space, with the number of attention heads set to eight (head dimension 16). The parameter count increases from 633K to 7,523K, by about a factor of 12.

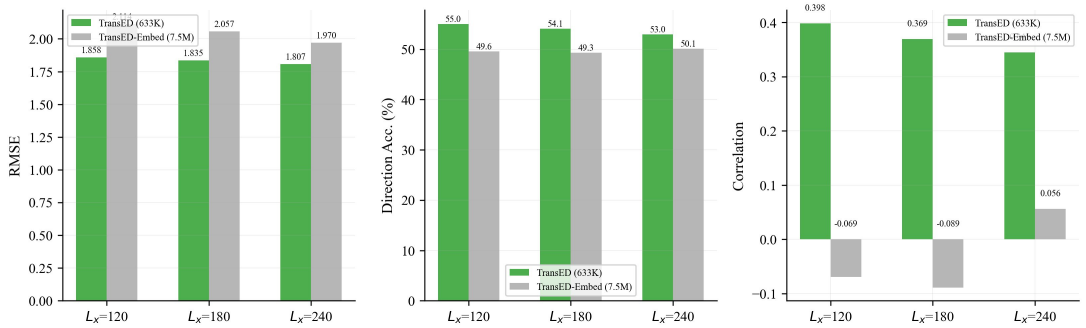

Figure 9. Comparison of TransED and TransED-Embed in RMSE, Directional Accuracy, and Correlation Coefficient at $L_x = 120, 180, 240$.

Figure 9 compares the two configurations at three L_x values. TransED-Embed is worse than the original TransED on all metrics and at all three windows. Taking $L_x = 120$ as an example:

Table 12. TransED vs. TransED-Embed at $L_x = 120$.

Metric	TransED	TransED-Embed	Change
RMSE	1.858	2.114	+13.7%
Directional accuracy	55.02%	49.58%	−5.44 pp
Correlation coefficient	0.398	−0.069	sign flipped
$\text{MASE}_{\text{naive}}$	0.973	1.175	crosses baseline ($<1 \rightarrow >1$)

At $L_x = 180$ and $L_x = 240$, the RMSE of TransED-Embed deteriorates by 12.1% and 9.0%, respectively, and the correlation coefficient at $L_x = 180$ falls to -0.089. Negative correlation means that the model fails to consistently track the direction of the true price changes. After the parameter count is scaled up by about 12 times, all metrics deteriorate. The mechanism behind this phenomenon is analysed in Section 6.2.

It should be noted that d_{model} and parameter count vary jointly in the TransED-Embed experiment; the two are confounded by design. The conclusion of this section should therefore be stated strictly as follows: under the current data scale, jointly enlarging d_{model} and parameter count leads to performance deterioration; the cause cannot be attributed to d_{model} alone. Disentangling these factors would require a controlled experiment that varies parameter count and d_{model} independently, which we leave for future work. This confound is discussed further in Section 6.

5.5 Interaction Effects of Observation Window Length and Training Cost

Figure 10 shows how the percentage gap between the RMSE of TransE and that of ARMAGARCH changes with L_x. The gap shrinks from 17.0% at $L_x = 120$ to 1.4% at $L_x = 260$ (the minimum within the tested range), and then rebounds to 3.1% at $L_x = 300$. TransE never reaches the RMSE level of ARMAGARCH within the entire tested range of L_x.

All deep learning models show non-monotonic dependence on observation window length, but their optimal windows and curve shapes differ markedly. This indicates an interaction effect between observation window length and model architecture. The so-called "optimal L_x" is model-specific, and no universal optimum exists.

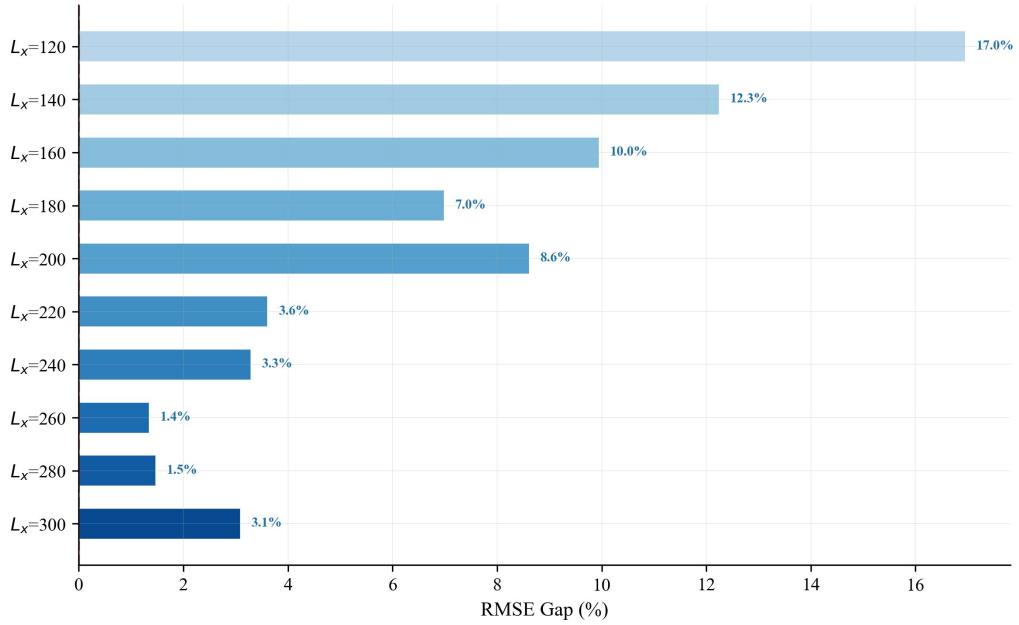

Figure 10. Percentage Gap in RMSE Between TransE and ARMAGARCH as a Function of L_x.

Table 13. Summary of Each Model's Response Pattern to L_x.

Model	Response shape	RMSE total variation	Notes
ARMAGARCH	Near-flat	7.7%	Least sensitive to L_x
TransE	First-decrease-then-increase	9.2%	Optimum near $L_x \approx 240$
TransED	U-shape	9.5%	Middle windows perform worse
BiLSTM	Irregular fluctuation	22.8%	Unstable pattern
TransED-Embed	–	3 points	Worst at all three windows

Apart from the response patterns to L_x, training cost is another factor to consider when selecting L_x in practice.

Figure 11 shows the training time of the four main models at $L_x = 120, 180, 240$. TransE has the longest training time, which grows rapidly with L_x: approximately 31 minutes at $L_x = 120$ and about 78 minutes at $L_x = 240$, owing to the $O(L^2)$ computational complexity of self-attention. ARMAGARCH does not require training in the deep learning sense, but re-fits on each test window at a cost of approximately three seconds per window, amounting to about 4.5 minutes per L_x configuration; this is roughly one order of magnitude faster than TransE at the same L_x, although ARMAGARCH runs on CPU while the deep learning models use MPS GPU acceleration. Extending L_x from 120 to 240 improves the RMSE of TransE by 8.1% but increases its training time by about 2.5 times. This performance–cost trade-off cannot be ignored when selecting the observation window length in practice.

Finally, Figure 12 presents the prediction visualisation of a representative test sample, offering a direct comparison between the predicted trajectories of the two best-performing models and the true closing prices.

Both models roughly track the direction and turning points of the true prices, although they differ in specific numerical values. This visualisation complements the quantitative conclusions above: even though ARMAGARCH dominates on all aggregated metrics, its predicted trajectory differs from that of

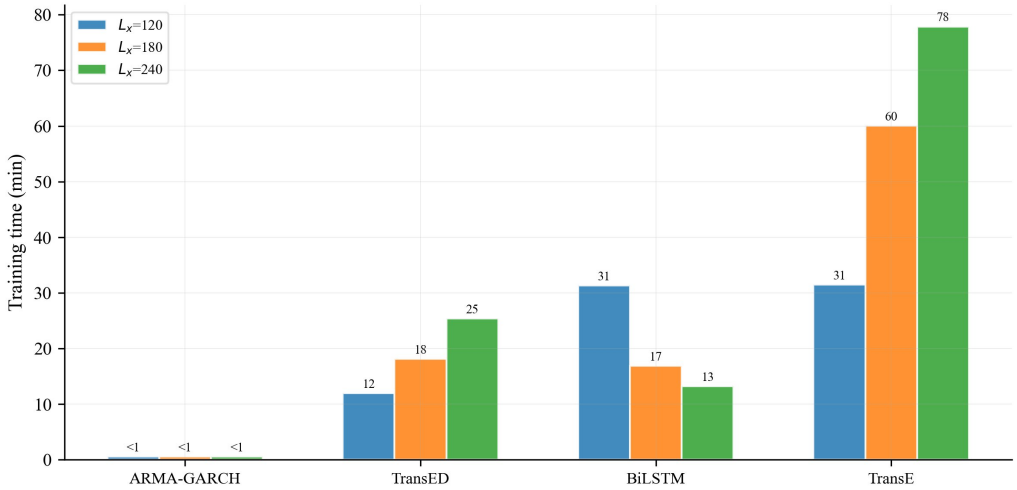

Figure 11. Training Time of the Four Main Models at $L_x = 120, 180, 240$.

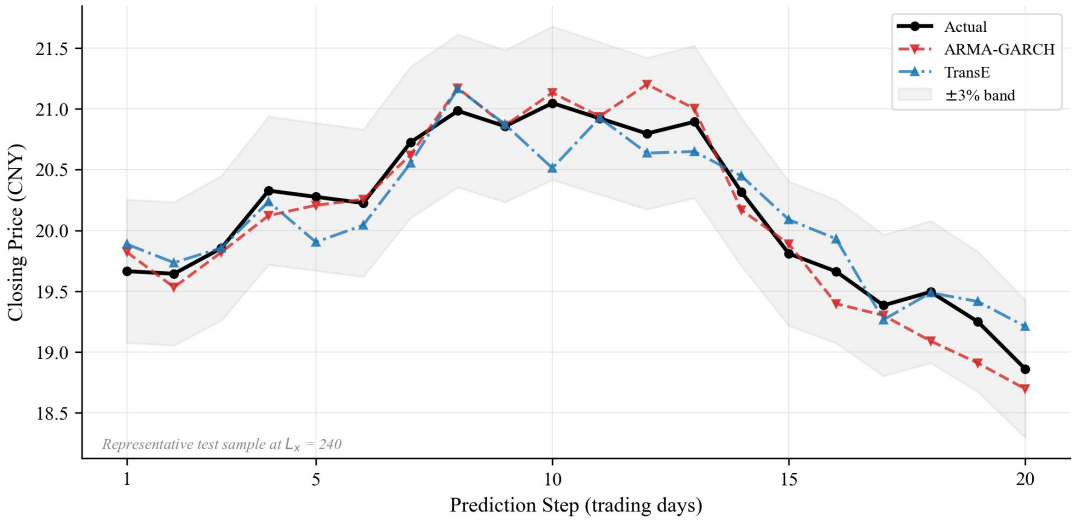

Figure 12. Predictions of ARMAGARCH and TransE on a Representative Test Sample ($L_x = 240$) Over the Next 20 Trading Days. The black line shows the true value; the light grey band marks the $\pm 3\%$ error interval.

TransE by only a modest margin at the single-sample level, and the difference mainly lies in average-level stability rather than in any single prediction.

6 Discussion

This section discusses the experimental results reported in Section 5. We first analyse possible causes of ARMAGARCH's competitiveness, then examine the mechanisms through which model complexity and observation window length affect predictive performance, then summarise the practical implications, and finally outline the limitations of this study.

6.1 Competitiveness of ARMAGARCH

Under our experimental setting, no deep learning model outperforms ARMAGARCH across the tested L_x range, as documented in Section 5.2. This result is in the same direction as the findings of Zeng, Chen, Zhang, and Xu (2023) on DLinear, who report on multiple time-series benchmark datasets that simple linear models can match or exceed sophisticated Transformer variants. Our contribution differs in two respects: first, we extend this observation from general time-series benchmarks to mid-term forecasting of Chinese A-shares, where retail-driven price dynamics differ from the multivariate sensor data used in Zeng et al.; second, our baseline is ARMAGARCH, a structural model that explicitly captures volatility clustering, rather than a linear benchmark. The result therefore suggests that the limited room for deep learning in this setting reflects the underlying signal-to-noise structure of daily-frequency stock prices, rather than model simplicity alone. We analyse four possible causes.

(1) Data scale. Each stock has 1,024 trading days of data, and the five stocks combined yield approximately 4,400 raw samples. Because the similar stocks are highly correlated within the same period, the number of effective independent samples is far lower than 4,400. Deep learning models with 543K–7,523K parameters face large estimation variance at this data scale, whereas ARMAGARCH has very few parameters and gains an advantage in estimation stability under small-sample conditions.

(2) Time scale of the forecasting target. We predict the relative return over the next 20 trading days. At the roughly one-month scale, stock prices are influenced by macroeconomic events, policy changes, and market sentiment, which are difficult to capture from historical prices and technical indicators alone. ARMAGARCH's modelling of recent autocorrelation and volatility clustering aligns with the stable regularities at this scale, while deep learning models, in trying to learn more complex patterns, may fit noise as signal.

(3) Differences in inference mode. ARMAGARCH re-estimates its parameters on each test window independently, which can be regarded as a simple form of test-time adaptation. Deep learning models use parameters fixed at the training stage to perform inference over the entire test set. For financial series whose target distribution drifts over time, the former naturally adapts to the local statistical properties within each window, while the latter needs stronger robustness to distribution shift to achieve the same effect.

(4) Match between model structure and window length. The RMSE of ARMAGARCH varies by only 7.7% across all windows, showing near-insensitivity to L_x. This is consistent with the fact that ARMA relies on recent autocorrelation and that GARCH focuses on recent conditional variance. The same property also explains why ARMAGARCH achieves the highest directional accuracy at short windows: at $L_x = 120$, the directional accuracy reaches 58.43%, because short windows give greater weight to recent information, and ARMA is more effective at modelling recent price momentum.

It should be noted that the experimental setting of this paper constitutes a specific constraint: a single target stock, $L_y = 20$, and approximately 4,400 raw samples. On larger datasets, under shorter forecasting horizons, or with multi-stock joint modelling, the relative performance of deep learning models may differ. The analysis in this section is limited to the current experimental setting and should not be generalised into a blanket claim that "deep learning is overall inferior to ARMAGARCH in stock price prediction."

6.2 Non-monotonic Relationship Between Model Complexity and Predictive Performance

The ablation study in Section 5.4.2 shows that, under the current data conditions, TransED-Embed is inferior to the original TransED on all metrics. We now offer a mechanism-level interpretation.

The training curve of TransED shows that the validation loss reaches its minimum after about 17–40 epochs, after which the training loss continues to decrease while the validation loss begins to rise—a typical sign of overfitting. When d_{model} is expanded from 12 to 128, the expressive space of the model grows substantially, and under limited data scale and a low target signal-to-noise ratio the overfitting is further aggravated. A smaller d_{model} may therefore have acted as a form of implicit regularisation under small-sample conditions.

It should be noted that d_{model} ($12 \rightarrow 128$) and total parameter count (633K $\rightarrow$ 7.5M) vary jointly in this experiment, and the two are not strictly separated by design. The conclusion of this section should therefore be stated strictly as follows: under the current data scale, jointly enlarging d_{model} and parameter count does not improve performance and instead leads to deterioration. This directional finding is consistent with the observation of Zeng et al. (2023) on DLinear.

6.3 Model-Dependent Effect of Observation Window Length

Table 13 summarises the response pattern of each model to L_x. We now interpret these differences in terms of the architectural properties of each model.

ARMAGARCH is the least sensitive to L_x, consistent with the fact that its autoregressive-conditional-heteroskedastic structure relies mainly on recent autocorrelation and gains little at more distant lags.

The first-decrease-then-increase pattern of TransE can be interpreted as an information–noise trade-off in the self-attention mechanism. In the early range, as L_x grows, the model extracts additional patterns from longer history. When L_x exceeds approximately 240, more distant data introduce noise that is weakly related to the prediction target and instead hurt performance. This trade-off provides empirical guidance for the window selection of Transformer-type models on financial series.

The U-shape of TransED may be related to the different optimisation difficulties of the encoder–decoder architecture under different input lengths. Short windows provide sparse information but are easy to fit. Long windows provide rich information sufficient to compensate for the encoding complexity. In the middle region, neither advantage is prominent.

The irregular fluctuation of BiLSTM is consistent with the information decay and gradient propagation difficulty of recurrent structures on long sequences.

The "optimal observation window length" is therefore model-specific, and no universal optimum exists. In practice, L_x should be determined for each specific model and dataset through systematic experimentation, rather than by transferring values from studies based on other models. For ARMAGARCH, the choice of L_x has a limited effect on performance. For TransE, the optimal L_x is near 240, but the training cost grows rapidly with L_x. For TransED with nhead $= 3$, the middle range ($L_x = 180$–220) corresponds to a region of higher RMSE and should be avoided; the nhead $= 6$ configuration does not exhibit the same pattern and reaches its best RMSE at $L_x = 220$.

6.4 Practical Implications

The results of this paper have the following practical implications for financial time series forecasting.

First, establish a statistical baseline. Under approximately four years of single-stock data (about 880 samples per stock, augmented to about 4,400 samples through similar-stock pooling), the RMSE of ARMAGARCH (1.569–1.689) is lower than that of all deep learning models. Before deploying a deep learning model, a necessary validation step is to establish ARMAGARCH or a comparable classical baseline and confirm that the deep learning model outperforms it.

Second, evaluation should jointly consider point forecast accuracy, directional accuracy, and improvement over the naive baseline. Most deep learning models under the current setting fail to beat the "tomorrow equals today" naive baseline, and only TransE and ARMAGARCH keep $\text{MASE}_{\text{naive}} < 1$ across all windows. BiLSTM achieves an RMSE close to TransE in some windows, but its directional accuracy stays within 49–53% and its $\text{MASE}_{\text{naive}}$ is generally above one, which makes it of limited practical use for decision-making. Model evaluation should not rely solely on the absolute value of RMSE; it should also confirm the improvement over the naive baseline and the stability of directional signals.

Third, both model capacity and window length need to be tuned per model. Enlarging the parameter count of TransED-Embed from 633K to 7,523K worsens the RMSE by 11.6%. The optimal L_x of TransE is near 240, and its performance rebounds at longer windows. When the data scale is limited, both model capacity and observation window length may have optimal values, and these optima are model-specific. They should be tuned systematically for each configuration rather than borrowed from other models' experience.

6.5 Limitations

This study has the following limitations.

(1) Single target stock. The experiments cover only Luoyang Molybdenum (603993). Differences across stocks in volatility level, trend strength, and sector membership may affect the relative performance of the models. Our conclusions should be understood under this constraint, and generalisation to other stocks or markets requires further validation.

(2) Fixed forecasting horizon. $L_y = 20$ corresponds to a mid-term forecast of approximately one month. Shorter forecasting horizons, for example $L_y = 3$ or 5, are closer to the decision frequency of some real-world trading strategies, and the relative performance of the models may change accordingly.

(3) Limited feature scope. The input features include only prices and technical indicators, and do not incorporate unstructured information such as news text, analyst reports, or market sentiment. In settings with richer information sources, the relative advantage of deep learning models, and of Transformer models in particular, may become more pronounced.

(4) Test set size and evaluation period. The test set is drawn from the target stock's most recent 10% of samples and contains 88 samples at $L_x = 120$, with longer L_x values corresponding to even fewer samples. Together with the temporal correlation between adjacent samples, this introduces non-trivial variance into the evaluation metrics themselves. We report results under a single random seed (42) and do not run repeated experiments under multiple seeds. In addition, the evaluation period may be affected by

specific market conditions, and model performance in periods of high volatility or frequent policy changes may differ from that in more stable periods.

(5) Limitations of the ablation and control design. Our ablation of TransED-Embed does not strictly isolate the effect of d_{model} alone, as d_{model} and parameter count vary jointly. More broadly, the three methodological components of our framework—similar-stock search, two-stage PCA–ICA dimensionality reduction, and the g_t decoder initialisation—are not independently ablated in the present study. We do not compare a five-stock merged setting against a single-stock training setting, nor do we isolate the contribution of the PCA–ICA pipeline or the g_t scheme relative to alternatives. The model ordering reported in this paper cannot therefore be strictly attributed to any single design choice. All deep learning models are also trained under a single loss function (MSE); alternatives such as Huber loss, quantile loss, or direction-aware losses may affect the relative ranking of the deep learning models and are not examined here. Independent ablations of these components, and of alternative loss functions, are left for future work.

7 Conclusion

Using the SSE 180 constituents of the Chinese A-share market as the stock pool and Luoyang Molybdenum (stock code 603993) as the target stock, this paper presents a systematic empirical study of how observation window length and model architecture affect multi-step stock price forecasting. The comparison covers six model configurations—TransED (with nhead = 3 and nhead = 6), TransE, BiLSTM, ARMAGARCH, and TransED-Embed—across eight to ten different observation window lengths.

The experimental results show that ARMAGARCH achieves the lowest RMSE across all ten test windows and attains the highest directional accuracy in seven out of eight common windows. This advantage is further confirmed by $\text{MASE}_{\text{naive}}$: most deep learning models fail to beat the random-walk naive baseline, and only TransE and ARMAGARCH keep $\text{MASE}_{\text{naive}} < 1$ across all windows. This finding echoes the observation of Zeng et al. (2023) on DLinear and provides new empirical evidence that simple models may outperform complex ones under certain conditions, now extended to the mid-term forecasting of Chinese A-shares. Observation window length and model architecture exhibit a clear interaction effect. ARMAGARCH is almost insensitive to L_x. TransE shows a first-decrease-then-increase pattern with its optimum near $L_x = 240$. TransED exhibits a U-shaped curve. BiLSTM fluctuates irregularly. Extending the internal dimension of the Transformer from 12 to 128 does not improve performance but worsens the average RMSE and pushes the correlation coefficient from positive to negative. In contrast, varying the number of attention heads from three to six under the $d_{\text{model}} = 12$ constraint has a negligible effect.

The contributions of this paper can be summarised along three dimensions. Methodologically, we propose (i) a multi-distance voting-based similar-stock search; (ii) a sample-level local normalisation scheme that avoids cross-window information leakage; (iii) a two-stage PCA–ICA dimensionality reduction adapted to the Transformer input; and (iv) a decoder-input design tailored to direct multi-step forecasting, in which the zero-padding of Informer and related models is replaced by the single-step return at the end of the observation window. Empirically, through the systematic experiment on observation window length, we document the competitiveness of ARMAGARCH relative to four deep learning models as well as the interaction effect between L_x and model architecture. On the evaluation side, we introduce

two baseline-referenced metrics, R^2_{hist} and $\text{MASE}_{\text{naive}}$.

The conclusions of this paper are constrained by certain conditions, discussed in detail in Section 6.5. Under these conditions, the empirical findings help delineate the practical scope of deep learning models in the mid-term forecasting of Chinese A-share prices. Future work can proceed along the following directions. (1) Repeat the experiments on multiple target stocks from different sectors to assess the generalisability of the above findings. (2) Shorten L_y (e.g., to three or five trading days) to examine the relative performance of deep learning models under shorter forecasting horizons. (3) Incorporate unstructured information such as news text and earnings-announcement sentiment to re-evaluate Transformer-type models in settings with richer information sources. (4) Systematically vary d_{model} while holding the parameter count approximately constant to isolate the effect of the internal dimension on performance. (5) Conduct a dedicated ablation on the similar-stock search step to quantify its contribution to downstream predictive performance.

References

Ballings, M., Van den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42(20), 7046–7056.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3), 307–327.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control* (5th ed.). Hoboken, NJ: Wiley.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.

De Bondt, W. F. M., & Thaler, R. (1985). Does the stock market overreact? *Journal of Finance*, 40(3), 793–805.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4), 987–1008.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25(2), 383–417.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2), 654–669.

Fons, E., Dawson, P., Zeng, X.-J., Keane, J., & Iosifidis, A. (2020). Evaluating data augmentation for financial time series classification. *arXiv preprint* arXiv:2010.15111.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

Huang, W., Nakamori, Y., & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10), 2513–2522.

Jegadeesh, N., & Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *Journal of Finance*, 48(1), 65–91.

Ji, Y., Luo, Y., Lu, A., Xia, D., Yang, L., & Liew, A. W.-C. (2024). Galformer: A Transformer with

generative decoding and a hybrid loss function for multi-step stock market index prediction. *Scientific Reports*, 14, 23762. https://doi.org/10.1038/s41598-024-72045-3

Kim, K.-J. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1–2), 307–319.

Kontopoulou, V. I., Panagopoulos, A. D., Kakkos, I., & Matsopoulos, G. K. (2023). A review of ARIMA vs. machine learning approaches for time series forecasting in data driven networks. *Future Internet*, 15(8), 255.

Li, M., Zhu, Y., Shen, Y., & Angelova, M. (2023). Clustering-enhanced stock price prediction using deep learning. *World Wide Web*, 26(1), 207–232.

Li, T., Liu, Z., Shen, Y., Wang, X., Chen, H., & Huang, S. (2024). MASTER: Market-guided stock Transformer for stock price forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(1), 162–170.

Li, W., Bao, R., Harimoto, K., Chen, D., Xu, J., & Su, Q. (2020). Modeling the stock relation with graph network for overnight stock movement prediction. *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 4541–4547.

Li, Y., Wang, X., & Guo, Y. (2024). CNN-Trans-SPP: A small Transformer with CNN for stock price prediction. *Electronic Research Archive*, 32(12), 6717–6732. https://doi.org/10.3934/era.2024314

Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4), 1748–1764.

Paparrizos, J., & Gravano, L. (2015). k-Shape: Efficient and accurate clustering of time series. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1855–1870.

Sarıkoç, M., & Celik, M. (2025). PCA-ICA-LSTM: A hybrid deep learning model based on dimension reduction methods to predict S&P 500 index price. *Computational Economics*, 65, 2249–2315. https://doi.org/10.1007/s10614-024-10629-x

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, 90, 106181.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2018). A comparison of ARIMA and LSTM in forecasting time series. *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications*, 1394–1401.

Taieb, S. B., & Atiya, A. F. (2016). A bias and variance analysis for multistep-ahead time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 27(1), 62–76.

Tan, L., Zhang, X., & Zhang, X. (2024). Retail and institutional investor trading behaviors: Evidence from China. *Annual Review of Financial Economics*, 16, 459–483. https://doi.org/10.1146/annurev-financial-082123-110132

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008.

Wang, X., Yang, K., & Liu, T. (2021). Stock price prediction based on morphological similarity clustering

and hierarchical temporal memory. *IEEE Access*, 9, 67241–67248. https://doi.org/10.1109/ACCESS.2021.3077004

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition Transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34, 22419–22430.

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023). Are Transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9), 11121–11128.

Zhang, C., Sjarif, N. N. A., & Ibrahim, R. (2024). Deep learning models for price forecasting of financial time series: A review of recent advancements: 2020–2022. *WIREs Data Mining and Knowledge Discovery*, 14(1), e1519. https://doi.org/10.1002/widm.1519

Zhou, H., Zhang, S., Peng, J., Zhang, W., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond efficient Transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12), 11106–11115.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022). FEDformer: Frequency enhanced decomposed Transformer for long-term series forecasting. *Proceedings of the 39th International Conference on Machine Learning*, 27268–27286.