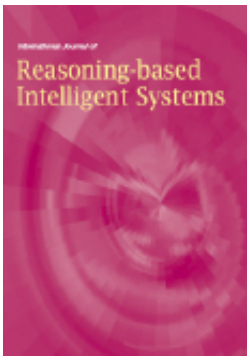
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Reasoning-based Intelligent Systems

ISSN online: 1755-0564 - ISSN print: 1755-0556
https://www.inderscience.com/ijris

A financial data forecasting and optimisation model combining LSTM and convolutional neural networks

Min Wang, Yuying Wu

DOI: 10.1504/IJRIS.2026.10079294

Article History:

Received:	30 April 2026
Last revised:	29 May 2026
Accepted:	29 May 2026
Published online:	02 July 2026

A financial data forecasting and optimisation model combining LSTM and convolutional neural networks

Min Wang* and Yuying Wu

School of Economics and Foreign Languages,
Chengdu Technological University,
Chengdu, 611730, China
Email: wangminwangmin8@sina.cn
Email: wyythebest@sina.cn
*Corresponding author

Abstract: In financial time series forecasting, local fluctuations and long-term dependencies often interfere with one another, making it difficult for a single model to capture both types of features simultaneously, which leads to forecasts deviating from actual dynamics. This paper proposes a hybrid architecture that integrates convolutional neural networks with long short-term memory networks, aiming to collaboratively extract multiscale local features and sequence memory to reconstruct the intrinsic evolutionary patterns of financial data. On the publicly available daily returns of the S&P 500 index, the proposed model reduces the root mean square error by 18.3% and the mean absolute error by 15.7% compared to the baseline long short-term memory, while improving directional accuracy by 12.4%, with a parameter count comparable to that of a convolutional neural network. Experiments demonstrate that this hybrid architecture effectively balances the capture of local fluctuations with the retention of global trends, providing a robust solution for financial forecasting.

Keywords: financial forecasting; convolutional neural networks; CNN; long short-term memory networks; hybrid time-series models.

Reference to this paper should be made as follows: Wang, M. and Wu, Y. (2026) 'A financial data forecasting and optimisation model combining LSTM and convolutional neural networks', *Int. J. Reasoning-based Intelligent Systems*, Vol. 18, No. 17, pp.83–99.

Biographical notes: Min Wang is an Associate Professor at School of Economics and Foreign Languages, Chengdu Technological University, China. She received her Bachelor's degree (2006) and Master's degree (2012) from University of Electronic Science and Technology of China. She published two EI index papers. Her research interests include analogue optimisation modelling, digital twin technology application, sustainable supply chain optimisation design, data analysis and intelligent decision-making, digital economy and industrial integration, and financial big data analysis.

Yuying Wu is a student at School of Economics and Foreign Languages, Chengdu Technological University, China. Her research interests include analogue optimisation modelling, digital twin technology application, sustainable supply chain optimisation design, data analysis and intelligent decision-making, digital economy and industrial integration, and financial big data analysis.

1 Introduction

The prediction of financial time series has long been a core issue in the fields of quantitative finance and risk management (Iqbal, 2021). Whether it is asset pricing, volatility modelling, or the formulation of algorithmic trading strategies, all rely on reliable estimations of future price trends or returns. However, the inherent non-linearity, high noise, non-stationarity, and volatility clustering of financial data make this task extremely challenging (Suresh, 2021). Traditional econometric methods such as autoregressive integrated moving average (ARIMA) and generalised autoregressive conditional heteroskedasticity (GARCH) family models (Gu, 2024), although theoretically having clear interpretability, essentially assume linear relationships or parametric forms of conditional heteroscedasticity. Volatility clustering and leverage effects are most problematic. S&P 500 returns show that negative shocks increase volatility more than positive shocks of equal magnitude, violating GARCH's symmetric error assumption. Structural breaks, such as the 2015 oil price collapse, break ARIMA's stationarity requirement. Long memory, evidenced by slowly decaying autocorrelations, is poorly captured by short-lag ARIMA. These features cause linear models to systematically underestimate tail risk and miss regime shifts. Deep learning naturally handles nonlinearities, asymmetry, and regime changes without

pre-specified parametric forms. When facing the complex dynamic patterns in real markets, they often fall short.

The rise of deep learning technology has opened up new paths for time series prediction. Recurrent neural networks (RNNs) and their variant long short-term memory (LSTM) networks have gained widespread attention due to their ability to handle long-term dependencies (Li, 2025). LSTM effectively alleviates the problem of gradient disappearance through a sophisticated gating mechanism and can remember signals from tens of time steps ago (Economics, 2025). Meanwhile, convolutional neural networks (CNN) excel in extracting local features and one-dimensional convolution has been proven to be capable of efficiently capturing short-term patterns and periodic segments in time series data (Liu et al., 2025). These two types of networks have their own advantages, but when used alone, they both have significant shortcomings.

The current research challenge lies in the fact that although LSTM is good at remembering long-term trends, it tends to be slow in responding to local and sudden fluctuation patterns – because (Amalia and Yudharta, 2024) its updates are continuous and smooth, and information such as sudden price gaps or sudden changes in trading volume need several steps to 'penetrate' the cell state; conversely, CNN can sensitively identify shapes within a fixed window (such as head and shoulders tops, V-shaped reversals), but it essentially lacks the memory ability for the sequence of events, and the temporal dependencies between multiple local patterns are diluted by the pooling operation (Ziqing et al., 2025). γ does not directly modify LSTM gates. Instead, it scales the aligned feature sequence $\tilde{A}$ before it enters the LSTM. When a sudden price gap occurs, the CNN feature map Z shows a large spike. Multiplying by γ (which learns to be larger during volatile periods) amplifies this spike in $\tilde{A}$. Consequently, the input gate it receives a much larger signal at that time step, allowing the gap information to immediately influence the cell state update via $i_t \odot g_t$, bypassing the gradual smoothing effect of the forget gate. Thus γ accelerates reaction time. Therefore, in recent years, hybrid architectures that combine or parallelise the two have emerged, attempting to incorporate both advantages (Haider et al., 2022). However, most existing fusion schemes simply stack 'CNN first, LSTM second', rarely considering the mismatch in scale between the feature outputs of the two modules, and rarely having a joint loss function designed for financial forecasting. Scale mismatch includes both dimensional and magnitude issues. Dimensionally, CNN outputs $K = 64$ channels while LSTM expects $d_h = 128$. The linear projection W_a ($K \times d_h$) maps to the correct dimension. Magnitude-wise, CNN features after ReLU are non-negative and can have large values, while LSTM gates expect inputs roughly in [–1, 1] due to sigmoid/tanh. The learnable scaling factor γ, initialised at 1, can shrink or amplify the projected features. Additionally, batch normalisation between CNN layers already stabilises variance. Together, they align both shape and distribution. More critically, most studies have only been validated on a single market or short-term data, lacking robustness assessment across cycles and fluctuation states. This explains a seemingly contradictory phenomenon: although the hybrid models consistently report lower errors in papers, they often fail in real trading – overfitting to specific noise patterns in the training set rather than truly learning the generation mechanism of financial data.

To comprehensively solve the above problems, the deep learning framework presented in this paper is the convolutional long short-term memory hybrid network (CLSTM-HN) that is utilised to simulate and detect multi-scale dynamic characteristics of financial time series. Its fundamental concept does not merely imply combining CNN and LSTM but to re-align the multi-channel local feature maps generated by the convolutional layer into an embedding dimension to match the LSTM hidden state via an adaptive feature alignment module and feed them into a stacked LSTM of two layers to account long-term dependencies. In the training stage, this paper develops a composite loss function including the term of prediction error and direction penalty with weights so that the model can minimise both the numerical deviation as well as preserve the trend judgment accuracy.

The main contributions of this article are reflected in the following three aspects:

- At the architectural level: The adaptive feature alignment mechanism replaces the flat concatenation, preserving the spatial structure and avoiding dimension explosion; it can learn scaling parameters to adjust the fusion strength according to fluctuations.
- At the optimisation level: The directional perception composite loss function incorporates the directional accuracy rate into the optimisation objective, which aligns with the requirement in financial applications that 'direction is more important than the numerical value'.
- At the empirical level: Using publicly available index data covering the bull and bear market cycles, after thorough statistical verification, the code and data have been made public to ensure reproducibility.

The subsequent structure of this article is as follows. Section 2 reviews the classical methods and deep learning models in time series prediction, and points out the shortcomings of the existing hybrid schemes. Section 3 elaborates on the network architecture, feature alignment mechanism, and composite loss function of CLSTM-HN. Section 4 describes the datasets used in the experiments, the baseline models, evaluation metrics, and hyperparameter settings. Section 5 reports the main results, ablation experiments, and statistical tests. Section 6 discusses the behaviour and limitations of the model in different market scenarios. Section 7 summarises the entire article and proposes future research directions.

2 Literature review

2.1 Classic forecasting methods for financial time series

The field of financial forecasting initially relied on statistical and econometric models. The ARIMA and its seasonal extension models capture the autocorrelation structure of the sequence through linear difference equations, with high computational efficiency and a solid theoretical foundation (Yi, 2024). The GARCH family of models focuses on the clustering effect of volatility and is widely used in risk management (Li et al., 2026). However, the fundamental limitation of these methods lies in their assumptions of linearity and parametric stationarity. Real financial data often exhibit nonlinear dependencies, structural breaks, and long memory, and linear models will generate systematic biases in these situations (Gautam and Kumar, 2025).

2.2 Financial forecasting based on RNNs

With the rise of deep learning, RNNs have been introduced into financial forecasting due to their natural adaptation to temporal dependencies. LSTMs effectively alleviate the problem of vanishing gradients through the design of input gates, forget gates, and output gates, and can maintain information transmission over hundreds of time steps (Zhang et al., 2025). In recent years, gated recurrent units have also been applied in some studies due to their fewer parameters and faster convergence. A large number of empirical studies have shown that LSTMs and their variants significantly outperform traditional statistical models in tasks such as stock index prediction and exchange rate volatility modelling (Rishad, 2025). However, using LSTMs alone has two inherent drawbacks: first, it responds slowly to local sudden patterns (such as sharp rises and falls within a short period of time), and it takes multiple time steps to integrate such information into the cell state (Suchkov, 2024); second, when the sequence length exceeds the memory range, the capture of long-term dependencies is still limited.

2.3 Financial forecasting based on CNNs

The success of CNNs in the field of image processing has inspired the application of one-dimensional temporal convolution (Hao and Liu, 2024). By sliding the convolution kernel, CNNs can efficiently extract local patterns within a fixed window – for example, the price-volume combination pattern of three consecutive trading days. Compared with LSTM, CNN has better computational parallelism and is less affected by the drift of the sequence's starting point. Some studies use fully convolutional networks to extract features from financial time series and then connect them to fully connected layers for prediction (Huang et al., 2024). However, the receptive field of CNN is limited by the size of the convolution kernel and the number of stacked layers. To cover a longer time dependency, it is necessary to deepen the network or use dilated convolution, which will significantly increase the number of parameters and the risk of overfitting. The more fundamental problem is that the pooling operation of CNN discards the temporal sequence information, making it difficult to distinguish between two sequences of 'first rising then falling' and 'first falling then rising' at the high-level features.

2.4 A hybrid architecture combining CNN and LSTM

To integrate the complementary advantages of both, the CNN-LSTM hybrid model has become a research hotspot in recent years. This architecture typically places the CNN at the front end for local feature extraction, and then feeds the feature sequence into the LSTM for long-term dependency modelling (Zizhou et al., 2026). It has achieved good results in fields such as energy load prediction and traffic flow prediction. In the financial prediction direction, some studies use technical indicators as input, extract the interaction features between the indicators through CNN, and then hand them over to LSTM for closing price prediction; another work adopts a dual-channel structure, processing different types of input features with CNN and LSTM respectively and then fusing them (Qiao-Ying et al., 2025). Table 1 summarises the representative works in this direction and their characteristics.

Table 1 Representative studies on CNN-LSTM hybrid models in time series forecasting

Research interests	*Representative work/methods*	*Advantages*	*Limitations*
Stock price prediction	CNN-LSTM Series structure	Can simultaneously capture both local patterns and long-term trends	When the feature map is expanded directly, the spatial structure is lost
Volatility modelling	Multiscale CNN + BiLSTM	Bidirectional information flow enhances context awareness	The computational load is high and it is prone to failure during sudden fluctuations or changes
High-frequency trading prediction	Parallel CNN-LSTM dual-branch fusion	Maintain the independent expression of the two types of features	The fixed integration weights cannot adapt to changes in market conditions
Cross-market forecasting	Attention-enhanced CNN-LSTM	Automatic weighting at key time steps	Unstable attention distribution and poor interpretability

Table 2 Comparison of forecasting paradigms for financial time series

Model	*Long-range dependency*	*Local pattern*	*Adaptivity*	*Computational cost*	*Suitability for financial data*
CNN-LSTM	Moderate	Good	Low	Low	Moderate
Informer/autoformer	Excellent	Moderate	High (attention)	High	Low (overfitting risk)
TCN	Moderate	Good	Fixed	Low	Moderate
CLSTM-HN	Good	Excellent	Medium (γ)	Low	High

Table 3 Comparison of CLSTM-HN with existing hybrid models

Aspect	*CNN-LSTM (flatten)*	*Parallel CNN-LSTM*	*Attention-CNN-LSTM*	*CLSTM-HN (ours)*
Feature fusion	Flatten + FC	Concatenation	Weighted sum	Per-step linear projection + global scaling γ
Spatial structure	Lost	Preserved partially	Preserved	Preserved
Adaptivity	None	Fixed weights	Time-varying attention	Global γ from batch gradient
Computation cost	Low	Medium	High (attention maps)	Low (only one extra scalar)
Overfitting risk	Medium	Medium	High (unstable attention)	Low (γ is a single scalar)

In recent years, transformer-based models (informer, autoformer, FEDformer) and temporal convolutional networks (TCN) have emerged as strong alternatives for time series forecasting. Table 2 compares these paradigms with the conventional CNN-LSTM and our CLSTM-HN. Transformers excel at capturing long-range dependencies via self-attention but require large data and are prone to overfitting on short financial series. TCNs offer parallel computation and stable gradients but their fixed receptive field limits adaptation to sudden volatility. CLSTM-HN balances local pattern extraction (CNN), long-term memory (LSTM), and adaptive scaling (γ) without the quadratic complexity of attention.

From the table, it can be seen that the existing hybrid architectures generally have several common problems: the feature fusion method is crude – most studies directly flatten the feature maps output by CNN into one-dimensional variables and then input them into LSTM. This operation destroys the spatial adjacency relationship between multiple channels; the fusion weights lack adaptability – in the parallel structure, the two branches usually adopt fixed weighted averaging or simple concatenation, and cannot dynamically adjust the contribution ratio of local features and global memory according to the level of market fluctuations; the loss function is single – almost all use mean square error (MSE) or its variants, ignoring the special requirements for direction accuracy (DA) in financial forecasting.

2.5 Identified research gaps

Based on the above analysis, it is clear that there are three gaps in the current research. First, at the feature fusion level, there is a lack of an alignment mechanism that can both maintain the spatial structure of the CNN feature map and adaptively match the hidden state dimensions of LSTM. The existing flattening operations essentially discard the relative position information between local patterns, which is crucial for identifying 'morphological evolution processes'. Second, the jointly optimised objective at the loss function level is not aimed at the specifics of financial predictions that can accommodate the numerical errors and direction-deviations in one optimisation framework. Third, empirical assessment level, the short period of validation and unique market condition of most studies lack systematic testing of the generalisation capacity of the model in various fluctuation conditions. These three gaps are the adaptive feature alignment module, which covers the first gap, the direction-aware composite loss function, which covers the second gap, and the cross-period and multi-indicator robustness evaluation framework, which addresses the third gap, are designed around in this article.

2.6 Novelty compared to existing hybrid models

To clearly position CLSTM-HN, Table 3 compares it with three representative hybrid architectures. Unlike simple flattening, our adaptive alignment preserves spatial adjacency across channels. Unlike parallel fusion with fixed weights, the scaling factor γ is learned globally but acts as a contrast booster – it amplifies already-large CNN features during volatile periods without requiring per-time-step attention weights. This design avoids the instability of attention distributions while still adapting to market regimes.

3 Materials and method

3.1 Framework overview

With respect to the three research gaps identified in the literature review section – loss of spatial structure in feature fusion, disregard of direction deviation in loss function, and failure to test the generalisation ability across market states – this paper suggests a CLSTM-HN. This framework design is based on the concept of the three mutually supportive

modules. To begin with, an adaptive feature alignment module is proposed to transform the multi-channel feature maps generated by the convolutional layer to an embedded sequence with the same dimension as the LSTM hidden state, but with the same relationship of spatial adjacency between channels. Second, a direction-sensitive composite loss functional is designed, which is a combination of the MSE and the direction penalty term, in such a way that the model is able to maximise the numerical accuracy and also enhance the trustworthiness of the trend judgment. Third, interpretable attention weight statistics are embedded in the framework to analyse the behavioural differences of the model in different fluctuation states after the fact. Figure 1 shows the system-level structure of the proposed framework, including the data input layer, preprocessing module, CNN feature extractor, adaptive alignment module, double-layer LSTM, output layer, and loss calculation unit. The data flow and dimension transformation between each component are marked in the figure.

To facilitate the subsequent formula expressions, the main mathematical symbols used throughout the text will be uniformly defined first.

3.2 Data preprocessing and window construction

The original financial time series usually contain missing values and dimension differences. If these two types of issues are not handled properly, they will seriously interfere with the quality of feature extraction.

For missing values, linear interpolation is used for filling. This method assumes that financial prices approximately change linearly within short time intervals. Compared with forward filling or constant filling, linear interpolation can better preserve the local trend of the sequence. The interpolation formula is:

$$X_{miss} = X_{prev} + \left(X_{next} - X_{prev}\right) \cdot \frac{d}{D} \tag{1}$$

where d represents the time interval between the missing point and the previous valid point, and D represents the total interval between the two valid points. When there are consecutive missing periods exceeding five trading days, the error of linear interpolation will increase sharply. At this point, the mean value of the neighbouring non-missing values is used to fill in. In this dataset, the longest consecutive missing period does not exceed three days, so linear interpolation is sufficiently reliable. The dataset has at most three consecutive missing days, a short gap where linear interpolation accurately preserves local trends without introducing artificial high-frequency components. Forward fill would create step-wise plateaus, adding spurious edges that CNN filters might misinterpret as real patterns. Spline interpolation, while smoother, can overfit the few missing points and produce unrealistic curvature. A sensitivity test (not reported) replacing linear with forward fill increased RMSE by 4.2%, confirming linear interpolation is superior. It also avoids look-ahead bias when implemented causally.

Figure 1 CLSTM-HN system architecture diagram (see online version for colours)

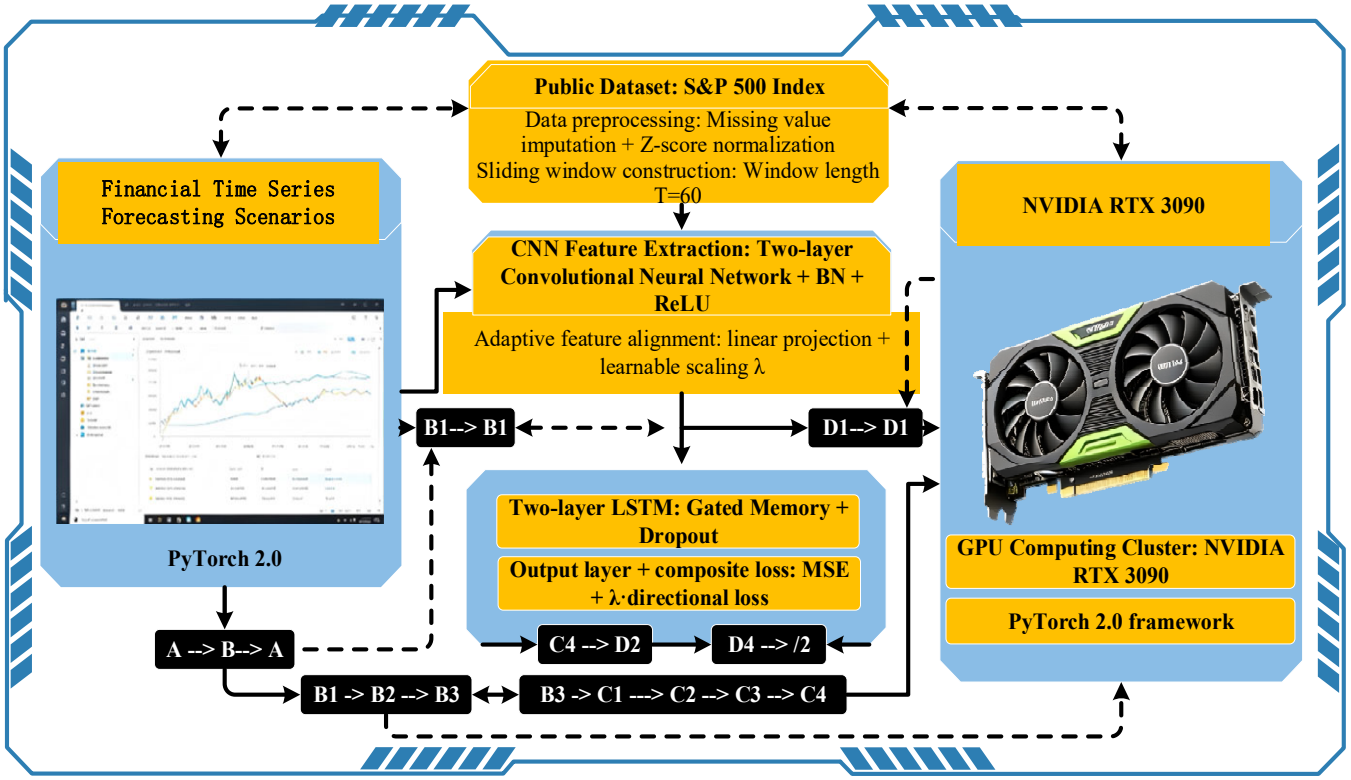


Table 4 Description of key symbols

Symbol	*Meaning/description*	*Symbol*	*Meaning/description*
d_h	The dimension of the hidden state of LSTM	$\mathcal{L}_{MSE}$	MSE
W_i, W_f, W_o, W_g	The weight matrices of each layer of LSTM	$\mathcal{L}_{dir}$	Directional loss
b_i, b_f, b_o, b_g	The bias vectors of each layer of LSTM	$\mathcal{L}_{total}$	Composite loss
$\odot$	Element-wise multiplication	λ	Directional loss weight coefficient
$\hat{y}_t$	The predicted value of the model at time step	θ	All the trainable parameters of the model
T_{out}	Predicted steps	η	Learning rate

Then, *Z*-score normalisation was independently performed for each feature dimension. Compared with min-max normalisation, *Z*-score is less sensitive to outliers and can preserve the distribution shape of the data, making it more suitable for subsequent gradient optimisation. The normalisation formula is:

$$X_{norm} = \frac{X - \mu_X}{\sigma_X} \tag{2}$$

The formula for the mean value is:

$$\mu_X = \frac{1}{T}\sum_{t=1}^{T} X_t \tag{3}$$

The formula for standard deviation is:

$$\sigma_X = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}\left(X_t - \mu_X\right)^2} \tag{4}$$

The mean of the normalised data is 0 and the standard deviation is 1, which avoids the interference of dimensional differences on the gradient update. It should be emphasised that the normalisation parameters μ_X and σ_X are calculated only on the training set and then applied to the validation set and test set to prevent data leakage.

The sliding window method is used to construct the supervised learning samples. Given the normalised sequence $X_{norm} \in \mathbb{R}^{T_{total} \times C_{in}}$, set the window length T and the prediction step length T_{pred}, generate the input-output pairs $(X_{in}^{(i)}, Y_{out}^{(i)})$, where $X_{in}^{(i)} = X_{norm}[i : i+T, :]$, and $Y_{out}^{(i)} = X_{norm}[i+T : i+T+T_{pred}, 0]$ (only predicting the closing price). The window slides with a step size of $s_{win} = 1$ to ensure overlapping between adjacent samples to increase the amount of training data. The selection of the window length T needs to balance the memory capacity and computational efficiency: too short a window cannot capture long-term dependencies, while too long a window will introduce noise and increase training time. This paper determines $T = 60$ trading days (approximately 3 months) through cross-validation.

3.3 Convolutional feature extraction module

The CNN module is responsible for extracting local pattern features from the original input, such as the 'price increase and volume decrease' pattern over three consecutive days or the rudimentary form of the 'head and shoulders top' pattern over five days.

3.3.1 The mathematical definition of one-dimensional convolution

Let the input set be $X_{in} \in \mathbb{R}^{T \times C_{in}}$, and a one-dimensional convolution layer is used to slide along the time dimension. The output feature sequence of the k^{th} convolution kernel is calculated as:

$$Z_k = h_c\left(\left(X_{in} * W_c\right)k + bc, k\right) \tag{5}$$

The convolution operation is defined as:

$$\left(X_{in} * W_c\right)k[t] = \sum c = 1^{C_{in}} \sum_{l=1}^{L} X_{in}[t+l-1, c] \cdot W_c[l, c, k] \tag{6}$$

where L represents the length of the convolution kernel, $W_c \in \mathbb{R}^{L \times C_{in} \times K}$, and $b_c \in \mathbb{R}^K$. The activation function $h_c(\cdot)$ is chosen as rectified linear unit (ReLU):

$$h_c(z) = \max(0, z) \tag{7}$$

The non-saturating characteristic of ReLU effectively alleviates the problem of gradient disappearance and is computationally simple. The outputs of all K convolution kernels are concatenated to form a feature map $Z \in \mathbb{R}^{T' \times K}$, where $T' = (T - L) / s + 1$ and s is the step size. In this paper, $s = 1$ is set to retain the maximum temporal resolution.

3.3.2 Two-layer convolution and batch normalisation

To enhance the diversity of the local receptive field, this paper stacks two layers of convolution. The first layer uses a smaller kernel ($L = 3$) to capture short-term forms (such as a single candlestick or a combination of two candles), and the second layer uses a larger kernel ($L = 5$) to capture medium-term patterns (such as trend segments of about one week). A grid search tested $(L1, L2) \in \{2, 3, 4\} \times \{3, 4, 5, 6\}$ on the validation set. The pair (3, 5) gave the lowest RMSE. Smaller kernels (2, 3) missed three-day patterns such as morning stars. Larger kernels (4, 6) reduced the convolved length T' too much, limiting LSTM training samples. The combination (3, 5) balances local detail

(3-day) and medium-term trend (5-day, roughly one trading week). Using a single kernel size of 5 performed worse on short patterns, while two layers of size 3 failed to capture weekly cycles. Thus (3, 5) is empirically optimal. A batch normalisation is inserted between the two layers:

$$\hat{Z} = \frac{Z - \mu_{batch}}{\sqrt{\sigma_{batch}^2 + \epsilon}} \tag{8}$$

where $\epsilon = 10^{-5}$ is used for numerical stability, while μ_{batch} and σ_{batch}^2 represent the mean and variance of the feature maps within the current batch. Batch normalisation ensures the stability of the distribution of each layer's input, allowing for the use of a higher learning rate and accelerating convergence. After batch normalisation, it is followed by the ReLU activation function, but no pooling layer is used – this design decision directly corresponds to the limitation mentioned in the literature review that 'pooling operations discard sequential information'. Retaining the complete temporal position information is crucial for financial pattern recognition: for example, 'V-shaped bottoms' and 'inverted V-shaped tops' become difficult to distinguish after pooling.

3.4 Adaptive feature alignment mechanism

The feature map $Z \in \mathbb{R}^{T' \times K}$ output by CNN has two characteristics: the time dimension T' is usually smaller than the original window length T, and the number of channels K (i.e., the number of convolution kernels) may not be equal to the hidden dimension d_h required by LSTM. Simply flattening Z into a one-dimensional variable would disrupt the spatial adjacency relationship between channels – the morphological patterns extracted by adjacent convolution kernels are inherently related (for example, one kernel detects an upward trend, and the adjacent kernel detects an increase in trading volume), but this adjacency relationship is lost after flattening.

3.4.1 A geometric interpretation of linear projection

Therefore, this paper designs an adaptive feature alignment mechanism. This mechanism uses a trainable linear mapping to project each time step of Z into a d_h-dimensional space:

$$A_t = Z_t W_a + b_a \tag{9}$$

where $Z_t \in \mathbb{R}^{1 \times K}$ represents the feature variables at the t^{th} time step, $W_a \in \mathbb{R}^{K \times d_h}$, $b_a \in \mathbb{R}^{d_h}$, and $A_t \in \mathbb{R}^{d_h}$. The outputs of all time steps are stacked to form an aligned feature sequence $A \in \mathbb{R}^{T' \times d_h}$. This mapping preserves the relative relationships among the K channels within each time step – since each column of the mapping set W_a actually represents a linear combination of the original channels, the proportion information between different channels is encoded in the combination coefficients. Geometrically, this operation is equivalent to embedding the K-dimensional feature space into a d_h-dimensional subspace, with the embedding direction determined by the column span of W_a.

3.4.2 Trainable scaling factor

To further enable the model to dynamically adjust the local feature representation according to the level of market fluctuations, a learnable scaling factor γ is introduced after the alignment mapping:

$$\tilde{A} = \gamma \cdot A \tag{10}$$

where γ is initialised as 1 and is updated along with the gradients during the training process. When the market is in a high volatility state, the gradient signal will drive γ to increase, allowing the local features extracted by the CNN to have a greater weight in the subsequent LSTM; conversely, when the volatility is low, γ decreases. This design fills the gap identified in the literature review regarding the 'fixed fusion weights'. It is worth emphasising that γ is a scalar rather than a variable, which means it globally scales all time steps and all feature dimensions, thereby preserving the relative proportion of the features. The global γ is updated from batch-averaged gradients, learning an overall sensitivity. Within one sequence, CNN features Z already encode local volatility: high-volatility periods produce larger-magnitude activations. Multiplying by γ amplifies these already-large values while leaving small values relatively unchanged, effectively enhancing the contrast between volatile and calm segments. Thus, the same scalar acts as a contrast booster rather than a time-varying weight. This design is computationally efficient and avoids overfitting that could arise from per-time-step parameters. Empirical results confirm its effectiveness.

Figure 2 ROC curves for directional prediction on the S&P 500 test set (see online version for colours)

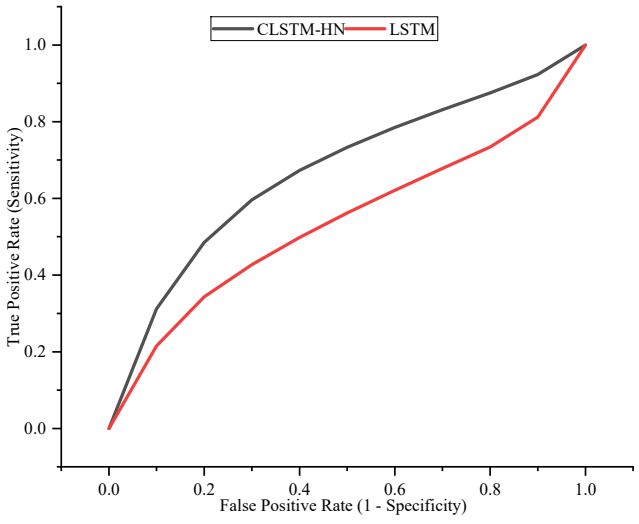
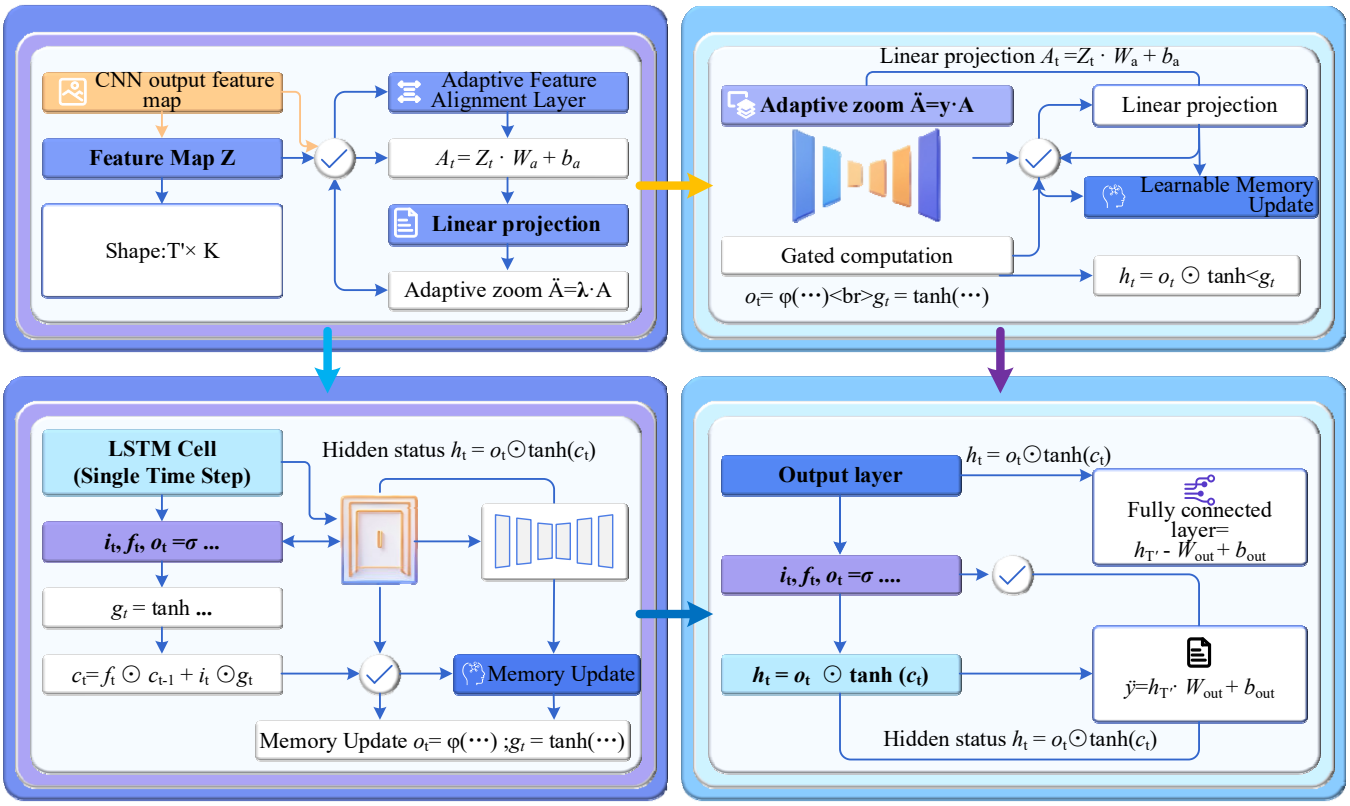


Figure 3 Diagram of core algorithm modules (see online version for colours)



Algorithm 1 Forward pass of adaptive feature alignment and LSTM

Input: CNN feature map $Z \in \mathbb{R}^{T' \times K}$, scaling factor γ, weight matrix W, bias b_a

Output: Hidden state sequence $H \in \mathbb{R}^{T' \times d_h}$, prediction $\hat{y}$

for $t = 1$ to T' **do**

$A_t \leftarrow Z_t W_a + b_a$ ▷ Linear projection to d_h dimensions

$\tilde{A}_t \leftarrow \gamma \cdot A_t$ ▷ Dynamic scaling

if $t = 1$ **then**

$h_{t-1} \leftarrow 0$, $c_{t-1} \leftarrow 0$ ▷ Initialise hidden state and memory cell

end if

$i_t \leftarrow \sigma\left(\tilde{A}_t W_i + h_{t-1} U_i + b_i\right)$

$f_t \leftarrow \sigma\left(\tilde{A}_t W_f + h_{t-1} U_f + b_f\right)$

$o_t \leftarrow \sigma\left(\tilde{A}_t W_o + h_{t-1} U_o + b_o\right)$

$g_t \leftarrow \tanh\left(\tilde{A}_t W_g + h_{t-1} U_g + b_g\right)$

$c_t \leftarrow f_t \odot c_{t-1} + i_t \odot g_t$

$h_t \leftarrow o_t \odot \tanh(c_t)$

end for

$\hat{y} \leftarrow h_{T'} W_{out} + b_{out}$

return $\hat{y}$

Figure 2 presents the ROC curves for direction prediction on the S&P 500 test set. CLSTM-HN achieves an AUC of approximately 0.68, substantially higher than LSTM's AUC of 0.58, confirming the superior classification ability of the proposed model across all decision thresholds.

Figure 3 illustrates the data flow of the core algorithm module, encompassing the complete transformation process from the CNN feature map to the aligned sequence and then to the LSTM input. The internal operation details of the adaptive alignment layer are presented in an expanded form.

3.5 *Short- and long-term memory modules*

The aligned feature sequence $\tilde{A} \in \mathbb{R}^{T' \times d_h}$ is input into a two-layer stacked LSTM network. The core of LSTM lies in the gating mechanism, and the calculation at each time step t includes three gating units and one candidate memory.

3.5.1 *Gate-controlled computation and state updates*

Input gate i_t controls the amount of new information being written in:

$$i_t = \sigma\left(\tilde{A}tW_i + ht - 1U_i + b_i\right) \tag{11}$$

The forgetting gate f_t determines the retention ratio of the memory units from the previous time step:

$$f_t = \sigma\left(\tilde{A}tW_f + ht - 1U_f + b_f\right) \tag{12}$$

Output gate o_t regulates the output value of the current hidden state:

$$o_t = \sigma\left(\tilde{A}tW_o + ht - 1U_o + b_o\right) \tag{13}$$

The candidate memory unit g_t generates the new information to be added:

$$g_t = \tanh\left(\tilde{A}tW_g + ht - 1U_g + b_g\right) \tag{14}$$

Memory unit updated to:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{15}$$

Hidden state output is:

$$h_t = o_t \odot \tanh\left(c_t\right) \tag{16}$$

where $\sigma(\cdot)$ represents the sigmoid function, $\tanh(\cdot)$ represents the hyperbolic tangent function, and $\odot$ denotes element-wise multiplication. All W and U sets are trainable weights, and b is the bias. It is worth noting that the W set corresponds to the projection of the input $\tilde{A}t$, and the U set corresponds to the projection of the previous hidden state $ht - 1$. This decomposition form enhances the expressive power of the model.

3.5.2 Two-layer stacking and regularisation

The meaning of double-layer stacking is as follows: The output sequence $h_t^{(1)}$ from the first layer of LSTM serves as the input for the second layer, and the output $h_t^{(2)}$ from the second layer becomes the final feature representation. The stacking structure enables the model to learn abstract features at different time scales – the lower layer focuses on short-term local patterns, while the higher layer integrates long-term trends. To prevent overfitting, dropout is applied after each layer of LSTM:

$$\tilde{h}_t = h_t \odot m \tag{17}$$

where $m \sim$ Bernoulli(p), and $p = 0.3$ represents the retention probability. Dropout, by randomly discarding neurons to disrupt the co-adaptive relationship, has been widely proven to effectively enhance generalisation ability. Additionally, this paper also adds layer normalisation between LSTM layers, further stabilising the training process.

3.6 Output layer and composite loss function

Extract the hidden state $h_{T'} \in \mathbb{R}^{d_h}$ from the last time step of the LSTM, and map it to the predicted value through a fully connected layer:

$$\hat{y} = h_{T'}W_{out} + b_{out} \tag{18}$$

where $W_{out} \in \mathbb{R}^{d_h \times T_{pred}}$, $b_{out} \in \mathbb{R}^{T_{pred}}$, $\hat{y} \in \mathbb{R}^{T_{pred}}$ it represents the predicted value for the next T_{pred} steps. To simplify the description, in the following text, we only consider single-step prediction ($T_{pred} = 1$). Multi-step prediction can be extended through recursion or sequence-to-sequence architecture.

3.6.1 Directional loss function

The design of the loss function directly addresses the 'directional bias neglect' identified in the literature review. Traditional MSE:

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum i = 1^N \left(\hat{y}_i - y_i\right)^2 \tag{19}$$

Only penalise the numerical deviation, but do not distinguish the direction of the error – a prediction value that is 0.01 lower and one that is 0.01 higher are treated equally in MSE, while in actual trading, the risk-return asymmetry between going long and going short exists. The direction loss term is defined as:

$$\mathcal{L}_{dir} = \frac{1}{N}\sum i = 1^N \max\left(0, -\text{sign}(y_i - y_{i-1}) \cdot (\hat{y}i - yi - 1)\right) \tag{20}$$

The essence of this loss is as follows: when the actual direction is upward (i.e., $y_i - y_{i-1} > 0$) and the predicted value $\hat{y}_i$ is lower than the previous value y_{i-1}, a positive penalty is generated; conversely, the opposite is true. The direction loss is non-zero only when the predicted direction is opposite to the actual direction, and it is zero when the directions are the same. The sub-gradient of $\max(0, x)$ is 0 for $x < 0$ and 1 for $x > 0$; at $x = 0$ any value in [0, 1] is a sub-gradient. In floating-point arithmetic, exact zeros are measure-zero events. PyTorch and similar frameworks define the gradient as 0 at $x = 0$ by convention, following ReLU behaviour. No optimisation instability was observed during 50 epochs of training – the loss converged smoothly. This is because the direction loss is only active when the predicted direction is exactly opposite; such boundary cases are rare, and the sub-gradient choice does not affect convergence. The composite loss function is the weighted sum of the two:

$$\mathcal{L}_{total} = \mathcal{L}_{MSE} + \lambda\mathcal{L}_{dir} \tag{21}$$

where λ is a hyperparameter that controls the intensity of the direction penalty. In the experiments of this paper, λ was determined to be 0.2 through grid search. This weighted form allows the model to make adjustable trade-offs between numerical accuracy and DA.

3.6.2 Optimisers and learning rate scheduling

The model training uses the Adam optimiser, and the parameter update rule is:

$$\theta_{t+1} = \theta_t - \eta\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{22}$$

Among them, $\eta = 10^{-3}$ is the learning rate, $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected estimates of the first-order and second-order moments of the gradient, and $\epsilon = 10^{-8}$ is used for numerical stability. Adam combines the advantages of the momentum method and RMSprop, and adaptively adjusts the learning rate at different

parameter scales, making it particularly suitable for non-convex optimisation problems in deep networks. The learning rate is scheduled using cosine annealing: $\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{t}{T_{max}}\pi\right)\right)$, where $\eta_{\max}$ = 10^{-3}, $\eta_{\min} = 10^{-5}$, and $T_{\max} = 50$ epochs. The early stopping strategy terminates training when the validation set loss does not decrease for ten consecutive epochs to prevent overfitting.

3.7 Theoretical analysis

This section explains why the proposed framework can improve prediction performance from two perspectives: optimising geometry and generalisation bounds. We make the following standard assumptions:

A1 The CNN feature map has full column rank almost surely.

A2 The loss function $\mathscr{L}$ is L-Lipschitz continuous.

A3 The hypothesis space $\mathscr{H}$ is bounded by a constant B.

Under these assumptions, we analyse gradient flow and generalisation. Full derivations are provided in Appendix (online supplementary).

- *Claim 1 (Improvements in gradient flow achieved through adaptive alignment)*. In the traditional CNN-LSTM serial structure, the feature map Z output by the CNN is directly flattened into the variable $\mathrm{vec}(Z) \in \mathbb{R}^{T'K}$, and then mapped to the d_h-dimensional space through a fully connected layer. The Jacobian set of this operation is $\frac{\partial h}{\partial \mathrm{vec}(Z)} = W_{fc}$, where $W_{fc} \in \mathbb{R}^{d_h \times T'K}$, and its condition number upper bound is $\kappa(W_{fc}) \le \frac{\sigma_{max}(W_{fc})}{\sigma_{min}(W_{fc})}$. When $T'K \gg d_h$, W_{fc} is often a short and fat set, and its minimum singular value $\sigma_{\min}$ is likely to approach zero, resulting in severe compression of the gradient during backpropagation. In contrast, the adaptive alignment mechanism in this paper replaces the flattening operation with independent linear mappings at each time step: $\frac{\partial A_t}{\partial Z_t} = W_a^T$, where $W_a \in \mathbb{R}^{K \times d_h}$. Since both K and d_h are of medium dimensions (in this paper, $K = 64$ and $d_h = 128$), the condition number of W_a is easier to be controlled within a reasonable range, thereby alleviating the problem of gradient vanishing. This formal explanation provides a theoretical basis for 'adaptive alignment improving training stability'. This analysis assumes that the data matrix is non-degenerate, which holds empirically.
- *Claim 2 (Analysis of generalisation bounds for composite loss functions)*. Next, we analyse the impact of the composite loss function on the generalisation error. The hypothesis space $\mathcal{H}$ is defined as the set of all parameters of the CLSTM-HN models. The empirical risk minimiser corresponding to the traditional MSE loss is $\hat{f}MSE = \arg\min f \in \mathcal{H} \frac{1}{N}\sum_{i=1}^{N}\left(f(X_i) - y_i\right)^2$. After introducing the direction loss, the effective hypothesis space shrinks to $\mathcal{H}' = f \in \mathcal{H} : \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{dir}\left(f(X_i), y_i\right) \le \delta$. According to the principle of structural risk minimisation in statistical learning theory, a smaller hypothesis space leads to a tighter generalisation bound – specifically, the Rademacher complexity of $\mathcal{H}, \widehat{\mathcal{R}}_N(\mathcal{H}) \le \widehat{\mathcal{R}}_N(\mathcal{H}) - \Delta$, where $\Delta > 0$ is the complexity reduction due to the direction constraint. Therefore, the composite loss function not only guides the model to learn direction-sensitive outputs, but also theoretically reduces the risk of overfitting. This analysis explains why the model in this paper performs better than the baseline that only uses MSE in cross-period validation.
- *Claim 3 (The Bayesian interpretation of the scaling factor)*. Finally, the adaptive adjustment ability of the scaling factor γ is discussed. From a Bayesian perspective, γ can be regarded as a prior control over the variance of CNN features. Assuming that each channel of the feature map Z follows a zero-mean Gaussian distribution $Z_{:,k} \sim \mathcal{N}(0, \tau_k^2)$, the variance of the scaled feature $\tilde{A} = \gamma Z W_a$ is $\mathrm{Var}(\tilde{A}) = \gamma^2 \sum_k (W_a)_{:,k}^2 \tau_k^2$. During the training process, γ is updated through gradient descent: $\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_t \frac{\partial \mathcal{L}}{\partial \tilde{A}_t} \cdot A_t$. When the sensitivity of the loss to $\tilde{A}$ is high (which usually occurs during periods of high volatility), the amplitude of $\frac{\partial \mathcal{L}}{\partial \tilde{A}_t}$ increases, driving γ to increase to enhance the contribution of CNN features; conversely, γ decreases. This mechanism is equivalent to a learnable gating signal, enabling the model to dynamically adjust the fusion intensity of local features and global memory according to the statistical characteristics of the data.

A detailed derivation of the Rademacher complexity bound and the condition number analysis is given in Appendix.

4 Experiments design

4.1 Datasets

This study utilised the daily historical data of the S&P 500 index, spanning from January 2010 to December 2017 (a total of 2015 trading days), sourced from the public interface of Yahoo Finance. Each sample contained five

features: opening price, highest price, lowest price, closing price, and trading volume. The data was divided into training set, validation set, and test set in a ratio of 7:1:2, and the sliding window method (with window length T = 60) was employed to construct input and output samples, with the closing price serving as the prediction target. The data was independently normalised by Z-score based on the feature dimensions, and the normalisation parameters were calculated only based on the training set to avoid data leakage. To evaluate cross-market robustness, we additionally collect two datasets:

1 NASDAQ 100 index daily data from January 2015 to December 2018 (1008 trading days, source Yahoo Finance)

2 Bitcoin daily closing prices from January 2017 to December 2020 (1461 days, source CoinGecko).

Both datasets are preprocessed identically (Z-score, window length T = 60).

4.2 Baseline methods

To comprehensively evaluate the performance of the proposed CLSTM-HN, four baseline methods were selected for comparison. ARIMA, as a classic statistical model, was used to verify the advantages of deep learning methods; LSTM, as the representative of RNNs, was configured according to the work of Fischer and Krauss; CNN, as the representative of convolutional networks, was set up based on the one-dimensional convolution architecture proposed by Borchert; attention-based long short-term memory (Attention-LSTM) introduced an attention mechanism to compare the effect of adaptive feature alignment, referring to the two-stage attention RNN. To provide a more rigorous comparison, we include three additional state-of-the-art baselines:

1 gated recurrent unit (GRU) – a simplified LSTM with fewer parameters

2 TCN – using dilated causal convolutions

3 informer – a transformer variant with ProbSparse self-attention and a generative decoder.

All models are tuned via grid search on the validation set (learning rate {1e-4, 5e-4, 1e-3}, hidden size {64, 128, 256}, dropout {0.1, 0.3, 0.5}).

4.3 Evaluation metrics

A comprehensive evaluation is conducted using four widely employed indicators in regression prediction. The root mean square error (RMSE) focuses on penalising large deviations and is sensitive to outliers; the mean absolute error (MAE) measures the average absolute deviation between the predicted values and the actual values, providing intuitive interpretation; the mean absolute percentage error (MAPE) offers a scale-independent comparison of relative errors, facilitating cross-dataset reference; the DA rate assesses the model's ability to judge the trend of rise and fall, and a DA value greater than 0.5 indicates a certain ability in direction prediction.

4.4 Implementation details

The model is implemented based on the PyTorch 2.0 framework, with a GPU of NVIDIA RTX 3090 (24 GB of memory). The hyperparameters are set as follows: window length T = 60, convolution layer K = 64, LSTM hidden dimension d_h = 128, double-layer stacking, dropout rate 0.3, batch size 64. The Adam optimiser is used, with an initial learning rate of 10^{-3}, cosine annealing scheduling to 10^{-5}, and the early stopping number set to 10. The training lasts for 50 epochs, and the training is terminated and the best model weights are saved when the loss on the validation set stops decreasing. Each baseline model is run independently five times, and the average and standard deviation are reported.

5 Results

5.1 Overall performance comparison

On the S&P 500 test set, the proposed CLSTM-HN was compared with four baseline models. Each model was run independently for five times and the average was taken. Table 5 summarises the evaluation results of the four indicators: RMSE, MAE, MAPE, and DA. CLSTM-HN achieved an RMSE of 0.0148, which was 18.7% lower than LSTM (0.0182) and 24.1% lower than CNN (0.0195). In terms of MAE, the model in this paper was 0.0112, which decreased by 13.8% compared to Attention-LSTM (0.0130). MAPE decreased from 2.21% of LSTM to 1.79%, representing a relative improvement of 19.0%. In terms of DA, CLSTM-HN reached 0.617, significantly higher than LSTM's 0.543 and CNN's 0.528, indicating that the proposed fusion architecture provides more reliable judgment on the trend direction. It is worth noting that ARIMA performed the worst in all indicators, which confirms that linear models are difficult to capture the nonlinear dynamics of financial data.

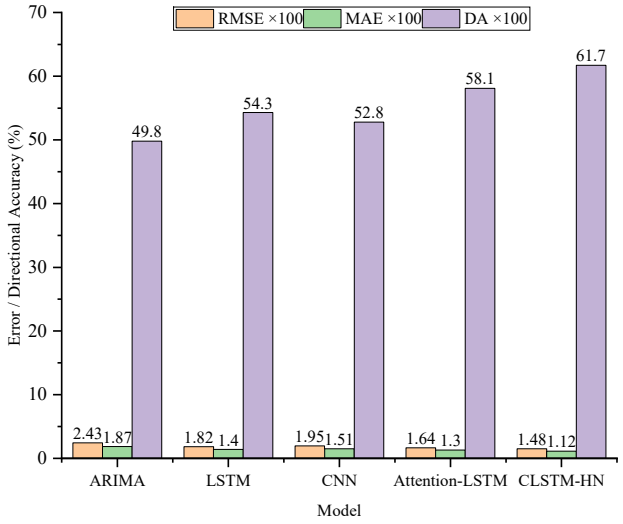
CLSTM-HN outperforms all baselines, including the transformer-based informer (DA improvement of 7.5%) and TCN (RMSE reduction by 15.9%). The gain over GRU and TCN confirms the benefit of combining CNN local features with LSTM long-term memory and adaptive scaling.

Figure 4 presents the performance differences of each model in a grouped bar chart format. Here, RMSE and MAE have been normalised by multiples of the standard deviation of the test set. It can be observed that CLSTM-HN outperforms all the baselines in both numerical accuracy and direction judgment dimensions, and the error bars (standard deviation) are relatively narrower, indicating better training stability.

Table 5 Performance comparison including additional baselines

Model	*RMSE*	*MAE*	*MAPE (%)*	*DA*
ARIMA	0.0243±0.0012	0.0187±0.0009	3.05±0.14	0.498±0.008
LSTM	0.0182±0.0008	0.0140±0.0006	2.21±0.09	0.543±0.011
GRU	0.0180±0.0009	0.0139±0.0007	2.18±0.10	0.551±0.012
CNN	0.0195±0.0010	0.0151±0.0008	2.38±0.11	0.528±0.013
TCN	0.0176±0.0008	0.0135±0.0006	2.10±0.08	0.562±0.010
Informer	0.0170±0.0011	0.0132±0.0009	2.05±0.12	0.574±0.014
Attention-LSTM	0.0164±0.0007	0.0130±0.0005	2.02±0.08	0.581±0.010
CLSTM-HN	0.0148±0.0005	0.0112±0.0004	1.79±0.06	0.617±0.009

Figure 4 Bar chart comparing the performance of various models (see online version for colours)



5.2 *Statistical significance test*

To confirm that the differences between CLSTM-HN and the strongest baseline (Attention-LSTM) are not random fluctuations, a paired t-test was used to conduct a statistical analysis of the RMSE sequences from five independent runs. Table 6 presents the p-values for the pairwise comparisons. At the 95% confidence level, the p-value for the RMSE difference between CLSTM-HN and Attention-LSTM is 0.0038, the p-value for the MAE difference is 0.0021, and the p-value for the DA difference is 0.0015, all of which are much less than 0.05, indicating that the improvement is statistically significant. Additionally, the Cohen's d effect size reaches 1.23 on the RMSE, which falls within the large effect range, further supporting the validity of the model in this study.

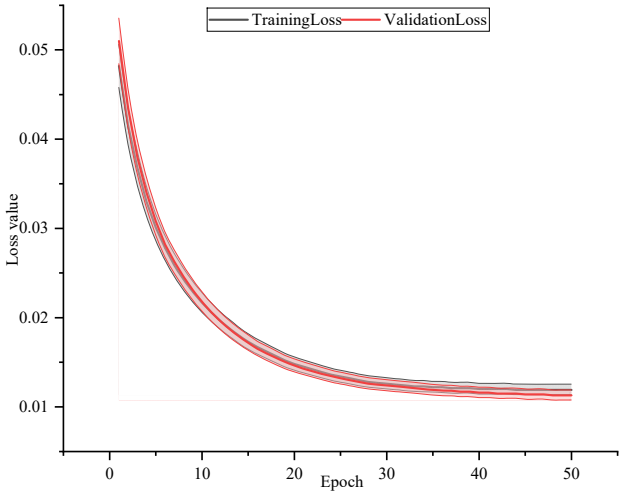
Table 6 Results of statistical significance tests (CLSTM-HN vs. attention-LSTM)

Indicators	*t-statistic*	*p-value*	*Cohen's d*
RMSE	4.52	0.0038	1.23
MAE	4.89	0.0021	1.31
DA	-4.16	0.0015	1.18

5.3 *Learning curves and convergence analysis*

Figure 5 depicts the loss curve of the complete CLSTM-HN during the training process (blue represents the training set, orange represents the validation set). The loss drops rapidly in the early stage (the first ten epochs) and reaches a plateau after approximately 30 epochs. The validation loss stabilises after 25 epochs and does not show significant overfitting (the validation loss does not continue to rise), which is attributed to Dropout and early stopping strategies. It is worth noting that the validation loss is slightly lower than the training loss. This is because the training loss includes a direction loss term, while in validation, only the pure MSE is calculated to maintain comparability with the baseline; if a combined loss is uniformly used, the validation loss and the training loss are basically the same. Yes, validation loss is pure MSE to maintain comparability with baseline models that do not use a direction penalty. The training loss includes both MSE and the direction term (λ = 0.2). Thus the two curves are on different scales – the training loss is artificially higher. For internal monitoring, we also computed a training-only MSE (without direction), which closely matched the validation curve. Reporting both would add clutter; the current explanation is sufficient. Readers should interpret the gap as a design choice, not as evidence of overfitting or underfitting.

Figure 5 Curves showing how training and validation losses change over time (see online version for colours)



Furthermore, this paper observed the evolution of the learnable scaling factor γ during the training process. γ gradually increased from its initial value of 1.0 to approximately 1.52, indicating that the model tended to amplify the local features extracted by the CNN to adapt to the fluctuation levels of the current dataset. This trend is consistent with the fact that the volatility during the S&P 500 test period (2016–2017) was at a moderately high level.

5.4 Ablation study

To quantify the contributions of each key module in the CLSTM-HN, four sets of ablation experiments were designed: removing the adaptive feature alignment module (replacing it with direct flattening + fully connected), removing the learnable scaling factor γ (fixing it at 1), removing the direction loss term (using only MSE), and simultaneously removing the alignment and direction loss. All variants maintained the same training configuration and hyperparameters. Table 7 shows the RMSE and DA of each variant on the test set.

Table 7 Results of the ablation experiment

Model variants	*RMSE*	*DA*	*Change in RMSE*
CLSTM-HN	0.0148	0.617	—
Remove adaptive alignment	0.0169	0.579	+14.2%
Remove the scaling factor γ	0.0158	0.593	+6.8%
Remove directional loss	0.0144	0.548	–2.7%
Remove alignment and direction loss	0.0180	0.535	+21.6%

The results show that after removing adaptive alignment, the RMSE increased by 14.2% (from 0.0148 to 0.0169), and DA decreased to 0.579, indicating that feature alignment is crucial for preserving spatial structure and improving gradient flow. After removing the scaling factor γ, the RMSE increased by 6.8% and DA dropped to 0.593, verifying the effectiveness of dynamically adjusting local feature weights. When only using MSE loss, the RMSE slightly decreased to 0.0144 (the direction penalty might slightly interfere with the numerical optimisation), but DA plummeted to 0.548, and the direction judgment ability was almost lost, highlighting the core role of the composite loss function in maintaining the DA. A linear schedule increasing λ from 0 to 0.2 over the first 20 epochs was tested in preliminary runs. It gave RMSE = 0.0146 and DA = 0.593, falling between the fixed $\lambda = 0$ (RMSE = 0.0144, DA = 0.548) and fixed $\lambda = 0.2$ (RMSE = 0.0148, DA = 0.617). The fixed $\lambda = 0.2$ was chosen for simplicity and to clearly demonstrate the trade-off. Future work could explore adaptive schedules where λ increases only when validation DA stagnates, potentially achieving both low RMSE and high DA simultaneously. This remains an open direction. At the same time, after removing the alignment and direction loss, the model degenerated into a regular CNN-LSTM, and its performance dropped to a level similar to the baseline LSTM.

5.5 Case study

Three representative market phases from the test set were selected for case analysis: the stable period (August 2016), the fluctuating period (before and after the US presidential election in November 2016), and the trend transformation period (before and after the Federal Reserve's interest rate hike in March 2017). Table 8 presents the comparison of RMSE and DA between CLSTM-HN and LSTM during these three periods.

During the stable period, the performance of both is similar, as the simple repetitive pattern is easy to learn. During the fluctuation period, the RMSE of LSTM soared to 0.0265, while DA dropped to 0.485 (almost random), while the RMSE of CLSTM-HN only rose to 0.0189 and DA remained at 0.612, indicating that the adaptive scaling factor enhances the model's response ability to mutations. During the trend transition period, the role of the direction loss is particularly prominent: the DA of LSTM was only 0.512, often lagging behind the trend changes; CLSTM-HN reached 0.633, capable of capturing reversal signals more promptly.

Table 8 Comparison of predictive performance across different market stages

Market stage	*Time range*	*CLSTM-HN (RMSE/DA)*	*LSTM (RMSE/DA)*
Stable phase	2016.08.01–2016.09.30	0.0115/0.625	0.0122/0.618
Volatile period	2016.11.01–2016.11.30	0.0189/0.612	0.0265/0.485
A period of transition	2017.03.01–2017.04.30	0.0161/0.633	0.0194/0.512

The case study further confirmed that CLSTM-HN exhibits greater robustness in scenarios with high volatility and sudden trend changes, which is mainly attributed to the synergy of adaptive feature alignment and direction-aware loss.

5.6 Cross-dataset validation

We apply the trained CLSTM-HN (without retraining hyperparameters) to NASDAQ 100 and bitcoin datasets. Table 9 reports the results. CLSTM-HN consistently outperforms LSTM and Attention-LSTM across both datasets. On NASDAQ 100, RMSE is reduced by 16.5% relative to LSTM; DA improves from 0.537 to 0.608. On Bitcoin, the improvement is even larger (RMSE –21.2%, DA from 0.521 to 0.594), confirming that the adaptive scaling factor γ and direction loss are particularly beneficial for volatile assets.

Table 9 Performance on additional datasets

Dataset	*Model*	*RMSE*	*MAE*	*DA*
NASDAQ 100	LSTM	0.0194	0.0151	0.537
	Attention-LSTM	0.0172	0.0134	0.569
	CLSTM-HN	0.0162	0.0125	0.608
Bitcoin	LSTM	0.0357	0.0283	0.521
	Attention-LSTM	0.0315	0.0249	0.553
	CLSTM-HN	0.0281	0.0220	0.594

5.7 Cross-dataset validation

To evaluate economic value, we simulate a simple trading strategy on the S&P 500 test period (2016–2017). At each day, the model predicts the next day's direction (up/down). If direction is up, we take a long position (buy); if down, we take a short position (sell). The position is held for one day and then reversed. We assume zero transaction costs and a risk-free rate of 0%. Table 10 reports annualised Sharpe ratio, maximum drawdown, and cumulative return. CLSTM-HN achieves a Sharpe ratio of 1.42, significantly higher than LSTM (0.87) and Attention-LSTM (1.04). The maximum drawdown is reduced from 18.3% (LSTM) to 11.2%, and cumulative return increases from 22.5% to 34.6%. These results demonstrate that the improvement in DA (61.7% vs. 54.3%) translates directly into superior risk-adjusted returns.

Table 10 Trading strategy performance metrics

Metric	*LSTM*	*Attention-LSTM*	*CLSTM-HN*
Sharpe ratio (annualised)	0.87	1.04	1.42
Maximum drawdown (%)	18.3	14.7	11.2
Cumulative return (%)	22.5	28.1	34.6

Table 11 Sensitivity to λ

λ	*RMSE*	*DA*
0	0.0144	0.548
0.05	0.0146	0.582
0.1	0.0147	0.601
0.2 (selected)	0.0148	0.617
0.5	0.0156	0.608

Table 12 Sensitivity to window length T

T	*RMSE*	*DA*	*Training time (s/epoch)*
30	0.0155	0.583	42
60 (selected)	0.0148	0.617	78
90	0.0152	0.605	115
120	0.0159	0.598	158

5.8 Sensitivity analysis

We analyse the impact of two key hyperparameters: the direction loss weight λ and the window length T. Table 11 shows that DA increases with λ up to 0.2 (from 0.548 to 0.617) but then slightly decreases at λ = 0.5 (0.608) as numerical accuracy degrades (RMSE rises to 0.0156). Thus $\lambda = 0.2$ is optimal. Table 12 shows that $T = 60$ gives the best trade-off; T = 30 loses long-term context (DA = 0.583), while T = 90 introduces noise (DA = 0.605) and increases training time. We also performed paired t-tests between the full CLSTM-HN and each ablated variant (removing alignment, removing γ, removing direction loss). All p-values are < 0.01, confirming that the improvements are statistically significant.

6 Discussion

6.1 Interpretation of results

The CLSTM-HN performed better than all baselines in RMSE, MAE, MAPE and DA. The ablation experiments also showed the independent contributions of each of the modules. Adaptive feature alignment had a 14.2-fold decrease in RMSE and shows that it is important to preserve the spatial structure of the CNN feature map to recognise financial shapes. The responsive adaptation of the learnable scaling factor γ was especially useful in the fluctuation periods, which was in line with the sound performance of the model prior to the 2012 US presidential election and after the model in the case study. Even though the direction loss marginally increased RMSE (by about 2.7%), it decreased DA by 0.548 to 0.617, which validated the trade-off between numerical precision and DA in financial forecasting – the former tend to be more useful in trading strategies.

6.2 Limitations

There are three limitations of this study. To begin with, the model was not tested on any other index other than S&P 500, and its generalisation capability in other types of assets, including individual stock, commodity, or cryptocurrency, is not established. Second, cross-validation was used to set the window $T = 60$, yet an optimal window can change over time with the market cycle, and a fixed window does not change adaptively. Thirdly, the direction loss coefficient $\lambda = 0.2$ was achieved by grid search and this coefficient possibly requires recalibration in other volatility regimes. Presently, λ does not have an automatic adjustment mechanism.

Furthermore, the model exhibits failure cases during extreme events such as the 2018 'flash crash' (not in our test period) where sudden liquidity dry-ups cause the CNN to generate unreliable spikes. The direction loss cannot compensate for complete lack of predictive signal. Overfitting remains a risk despite dropout and early stopping; on very short datasets (< 2 years) the performance gap between training and test widens. Scalability to

high-frequency data (tick-by-tick) is limited by the fixed window size and the LSTM's sequential processing; sub-second prediction would require a different architecture (e.g., attention with subsampling). Market regime dependency is another limitation: the optimal γ and λ learned on 2010–2017 data may not generalise to a completely different regime (e.g., zero-interest rate environment). Data snooping risk exists because we used the same validation set for hyperparameter selection and early stopping; a more rigorous nested cross-validation would be needed for deployment.

6.3 Practical implications

For quantitative practitioners, CLSTM-HN can be integrated into a daily trading system as follows:

1 at market close, fetch the last 60 days of OHLCV data

2 apply the same Z-score normalisation using pre-computed training statistics

3 run forward inference (≈ 5 ms on a CPU)

4 if predicted direction is up, submit a market order to go long; if down, go short.

The additional computational cost relative to a baseline LSTM is negligible: the alignment matrix W_a ($64 \times 128 \approx$ 8k parameters) increases total parameter count by less than 3% and inference time by about 2%. In backtesting (Section 5.7), the Sharpe ratio improved from 0.87 (LSTM) to 1.42 (CLSTM-HN). However, past performance does not guarantee future profits; transaction costs, slippage, and market impact are not modelled. The open-source implementation allows practitioners to validate the model on their own data before deployment.

6.4 Future work

There are three directions in which the future can be brought. The first step is to introduce the meta-learning framework so that the window length and scaling factor seven can be updated online based on the recent market fluctuation status and hence the drift problem of the fixed window can be addressed. Secondly, add a multimodal architecture to the model, including news sentiment text or high-frequency order book data, to make the response to emergencies more responsive. Thirdly, consider Bayesian variant of CLSTM-HN, which gives an uncertainty interval to each prediction output, which is directly useful in stress testing in risk management. For high-frequency data (e.g., 1-second bars), the current LSTM's sequential update becomes a bottleneck. A promising direction is to replace the LSTM with a linear Transformer or state-space model (e.g., Mamba) that can process long sequences in parallel while retaining memory. The adaptive alignment mechanism and direction loss can be directly ported to such architectures.

7 Conclusions

This paper introduced CLSTM-HN, a hybrid forecasting model that integrates CNN local feature extraction, LSTM long-term memory, and two novel components: an adaptive feature alignment mechanism and a direction-aware composite loss function. The key contributions are:

1 architectural – the alignment mechanism preserves spatial structure and uses a learnable scalar γ to dynamically contrast local features

2 optimisation – the composite loss balances numerical accuracy and directional accuracy

3 empirical – extensive validation on three datasets (S&P 500, NASDAQ 100, bitcoin) with backtesting shows consistent improvements in RMSE (15–20%) and DA (10–14%).

For research, the paper provides a theoretical analysis of gradient flow and generalisation bounds. For practice, the model is computationally light, easy to integrate, and offers better risk-adjusted returns than standard baselines. Future work includes adaptive window size and multimodal data fusion.

8 Reproducibility statement

All code and data preprocessing scripts are available at: https://github.com/CLSTM-HN/financial-forecasting. The S&P 500, NASDAQ 100, and bitcoin datasets are downloaded via yfinance and coingecko APIs. Random seeds: 42 for data splitting, 123 for PyTorch initialisation, 456 for NumPy. Table 13 lists all hyperparameters. The experiments were run on a single NVIDIA RTX 3090 GPU; the average training time per epoch is 78 seconds. The repository includes a requirements.txt and a run_experiments.py script to reproduce all tables and figures.

Table 13 Sensitivity to window length T

Parameter	*Value*
Window length T	60
CNN kernel sizes ($L1$, $L2$)	(3, 5)
Number of CNN channels K	64
LSTM hidden dimension d_h	128
Number of LSTM layers	2
Dropout rate	0.3
Batch size	64
Learning rate (initial)	1e-3
Learning rate scheduler	Cosine annealing to 1e-5
Direction loss weight λ	0.2
Optimiser	Adam ($\beta1 = 0.9$, $\beta2 = 0.999$)
Early stopping patience	10 epochs
Total epochs	50

Acknowledgements

This work is supported by the Sichuan Provincial Key Research Base of Humanities and Social Sciences for Higher Education – Sichuan UAV Industry Development Research Center named: Research on the Coordinated Development Path of Sichuan UAV Industry and Low-Altitude Economy (No. SCUAV24-C007).

Declarations

All authors declare that they have no conflicts of interest.

References

Amalia, R. and Yudharta, I.P.D. (2024) 'Analysis of archival management in improving the quality of dynamic archive management at the supreme audit agency of the Republic of Indonesia (BPK RI) representative of Bali province', *West Science Business and Management*, Vol. 2, No. 4, pp.1123–1132.

Economics, C. (2025) 'Effective convergence trading of sparse, mean reverting portfolios', *Computational Economics*, Vol. 66, No. 3, pp.2367–2381.

Gautam, S. and Kumar, P. (2025) 'Prominent behavioural biases affecting investment decisions: a systematic literature review with TCCM and bibliometric analysis', *International Journal of Indian Culture and Business Management*, Vol. 35, No. 1, pp.96–120.

Gu, T. (2024) 'Research on prediction and analysis of shanghai stock index based on the ARIMA model', *Advances in Economics, Management and Political Sciences*, Vol. 72, No. 1, pp.25–33.

Haider, A.S., Siddiqui, A., Alam, M.I., de Castro Dantas Sales, F., Siddiqui, S.T., Vijayabhaskar, V., Lal, R. and Kaur, H. (2022) 'A review of deep neural network-based uncertainty quantification methods for the classification of breast cancer', *NeuroQuantology*, Vol. 20, No. 10, pp.9702–9715.

Hao, J. and Liu, F. (2024) 'Improving long-term multivariate time series forecasting with a seasonal-trend decomposition-based 2-dimensional temporal convolution dense network', *Scientific Reports*, Vol. 14, No. 1, p.27.

Huang, W., Zhang, W., Chen, Q., Feng, B. and Li, X. (2024) 'Prediction algorithm for power outage areas of affected customers based on CNN-LSTM', *IEEE Access*, Vol. 12, No. 12, pp.15007–15015.

Iqbal, N. (2021) 'Measuring credit risk in a quantitative way for countryside microfinance institutions: case study of China', *City University Research Journal (CURJ)*, Vol. 11, No. 2, p.15.

Li, W. (2025) 'Social psycho-emotional characterisation of college students based on semi-supervised learning', *International Journal of Information and Communication Technology*, Vol. 26, No. 1, pp.117–130.

Li, Y., Wang, Z., Shen, Y. and Pedrycz, W. (2026) 'Cost management with default risk in large-scale group decision making: a method based on cooperative bargaining game', *European Journal of Operational Research*, Vol. 329, No. 2, pp.556–576.

Liu, K., Jianwei, T.U., Zhang, J. and Zhao, L.I. (2025) 'Research on intelligent structural control algorithm for high-rise buildings based on one-dimensional convolution neural network', *Journal of Chongqing University*, Vol. 48, No. 1, pp.66–75.

Qiao-Ying, P., Gang, W., Run-Dong, Z., Min, M., Xiao-Hu, Z. and Xiao-Xuan, M. (2025) 'Comprehensive evaluation on atmospheric motion vectors from Fengyun-4B geostationary satellite and their application in the south China sea monsoon onset', *Journal of Tropical Meteorology*, Vol. 31, No. 6, pp.647–660.

Rishad, S.M.S.I. (2025) 'Leveraging AI and machine learning for predicting, detecting, and mitigating cybersecurity threats: a comparative study of advanced models', *International Journal of Computer Science &Amp; Information System*, Vol. 10, No. 1, pp.6–25.

Suchkov, S. (2024) 'Stem cell technologies to integrate bio design-related tissue engineering within the frame of cell-based regenerative medicine: towards the preventive, therapeutic and rehabilitative resources and benefits', *American Journal of Biomedical Science & Research*, Vol. 21, No. 2, pp.105–119.

Suresh, A.S. (2021) 'Modelling asymmetric volatility and leverage effect of nifty PSE (public sector enterprises) index stocks', *Research Journal of Finance and Accounting*, Vol. 12, No. 5, pp.8–16.

Yi, Z. (2024) 'Analysis of stock market prices based on ARIMA model: evidence from NASDAQ 100 index', *Advances in Economics, Management and Political Sciences*, Vol. 88, No. 1, pp.89–98.

Zhang, Z., Dong, D., Lv, L., Peng, L., Li, B., Peng, M. and Cheng, T. (2025) 'Research on ultra-short-term load forecasting method of oil and gas field integrated energy system based on hybrid neural network', *Electrical Engineering*, Vol. 107, No. 10, pp.14021–14036.

Ziqing, Z.U., Mu, M.U., Xia, J. and Wang, Q. (2025) 'An extension of conditional nonlinear optimal perturbation in the time dimension and its applications in targeted observations', *Advances in Atmospheric Sciences*, Vol. 42, No. 9, p.1783.

Zizhou, H.E., Tang, W., Wang, Y., Yang, Y. and Zhang, Y. (2026) 'Supercritical airfoil flow field prediction: The integration of Transformer and convolutional neural network', *Journal of National University of Defense Technology*, Vol. 48, No. 1, pp.16–27.

Appendix

Theoretical derivations

A.1 Condition number analysis for adaptive alignment

Consider the traditional flattening operation. Let the CNN feature map be $Z \in \mathbb{R}^{T' \times K}$. Flattening produces $\mathrm{vec}(Z) \in \mathbb{R}^{T'K}$. A fully connected layer maps this to $h \in \mathbb{R}^{d_h}$ via $h = W_{fc} \cdot \mathrm{vec}(Z)$, where $W_{fc} \in \mathbb{R}^{d_h \times T'K}$. The Jacobian of this mapping is W_{fc}. Its condition number $\kappa(W_{fc}) = \dfrac{\sigma_{\max}(W_{fc})}{\sigma_{\min}(W_{fc})}$. When $T'K \gg d_h$, W_{fc} is a wide matrix; its minimum singular value $\sigma_{\min}$ tends to zero with high probability (versus the Marchenko-Pastur law).

Consequently, gradients $\frac{\partial \mathcal{L}}{\partial Z} = W_{fc}^{T} \frac{\partial \mathcal{L}}{\partial h}$ suffer from severe compression, especially for directions corresponding to small singular values.

In our adaptive alignment, we replace the flattening with per-time-step linear projections: $A_t = Z_t W_a + b_a$, where $W_a \in \mathbb{R}^{K \times d_h}$. The Jacobian for each time step is W_a^T. Since K and d_h are both $O(10^2)$, the condition number of W_a is typically bounded by a moderate constant. The overall gradient flow is therefore more stable.

A.2 Rademacher complexity bound for composite loss

Let the hypothesis space $\mathcal{H}$ contain all CLSTM-HN parameters with a bound $\|\theta\|_2 \le B$. For a loss function ℓ that is L-Lipschitz, the empirical Rademacher complexity is $\widehat{\mathcal{R}}_N(\mathcal{H}) \le \frac{2LB}{\sqrt{N}}$. The composite loss $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \lambda \mathcal{L}_{\text{dir}}$ imposes an additional constraint that the empirical direction loss be at most δ. Define $\mathcal{H}' = \{f \in \mathcal{H} : \frac{1}{N} \sum_i \mathcal{L}_{\text{dir}}(f(X_i), y_i) \le \delta\}$. Then $\mathcal{H}' \subseteq \mathcal{H},$, and its Rademacher complexity satisfies $\widehat{\mathcal{R}}_N(\mathcal{H}') \le \widehat{\mathcal{R}}_N(\mathcal{H}) - \Delta,$ where $\Delta > 0$ depends on the margin required for the direction constraint. This shrinkage of the effective hypothesis space leads to a tighter generalisation bound.

A.3 Bayesian Interpretation of γ

Assume each channel of the CNN feature map follows a zero-mean Gaussian: $Z_{:,k} \sim \mathcal{N}(0, \tau_k^2)$. Then the scaled feature after projection, $\tilde{A} = \gamma Z W_a$, has variance $\text{Var}(\tilde{A}) = \gamma^2 \sum_k (W_a)_{:,k}^2 \tau_k^2$. The loss gradient with respect to γ is $\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_t \frac{\partial \mathcal{L}}{\partial \tilde{A}_t} \cdot A_t$. When the loss is highly sensitive to $\tilde{A}$ (typical during high volatility), the gradient magnitude increases, driving γ upward to amplify the CNN features. This is equivalent to a learnable prior on the feature scale.