Tutorial for Time Series Data Prediction through Machine Learning Methods

Student: 廖添富(P13942A09)

Graduate Institute of Communication Engineering

National Taiwan University

Table of Contents

Chapter 1: Introduction:

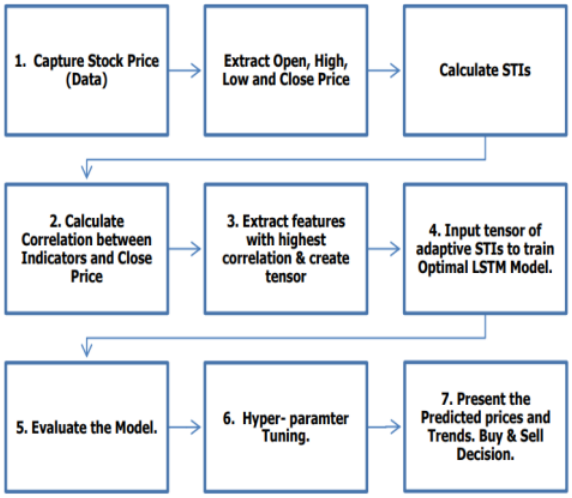
In recent years, artificial intelligence has become a popular technology applied in many fields to improve efficiency and accuracy. Time series (TS) data prediction is one of the areas where it has been widely used, especially in the stock market. Many studies have shown that machine learning methods can outperform traditional statistical models such as ARIMA, Random Forest, and SVM to achieve higher prediction accuracy. Among these methods, the LSTM model is one of the most popular, as it can capture the dependence between past and future data, providing a solid baseline of accuracy.

To gain more benefits from different types of models, hybrid models have been proposed. These combine different models at various stages and have shown slight improvements in prediction results. In addition, data preprocessing plays an important role before training the model. Several new approaches have been suggested to improve prediction performance, and training with reference data can also help by incorporating it into the training process.

Chapter 2: Single model

2.1 Technical Indicators: A Predictive Model using Optimal Deep Learning

Agrawal, Manish, Asif Ullah Khan, and Piyush Kumar Shukla [2] propose a Framework [Figure 1-1] that use optimal LSTM model and incorporating technical indicators [Figure 1-2] into the training sequence to achieve more accurate predictions.



[Figure 1-1] Proposed Framework for Stock Trends Forecasting. [2]

Stock Technical Indicators (STIs)	Mathematical Formula
Relative Strength Index (RSI)	$RSI = 100 - \left[\left(\frac{100}{1+\left(\frac{AG}{AL}\right)}\right)\right]$
Moving Average (MA) of n days	$\frac{1}{n}\sum_{i=1}^{n} C_i$
Stochastic Oscillator (%K)	$\%K = \left[\frac{C_t - Lp}{Hp - Lp}\right] * 100$
William (%R)	$\%K = \left[\frac{Hp - C_t}{Hp - Lp}\right] * 100$
Exponential Moving Average (EMA)	$EMA_t = C_t\left(\frac{2}{T+1}\right) + EMA_{t-1}\left(1 - \frac{2}{T+1}\right)$
Moving Average Convergence Divergence (MACD): most common is 12/26 MACD	$MACD = [(12 - day\ EMA) - (26 - day\ EMA)]$

Where Hp and Lp are highest and lowest prices in the last **p** days, respectively and C_t is the current price of the day under consideration. **AG** is Average Gain and **AL** is Average Loss.

[Figure 1-2] Stock Technical Indicators. [2]

2.1.1 Technical indicator

The first step is to obtain the technical indicator using the formula shown in [Figure 1-2]. Next, compute the correlation between the indicator and the closing price as [Figure 1-3] shown.

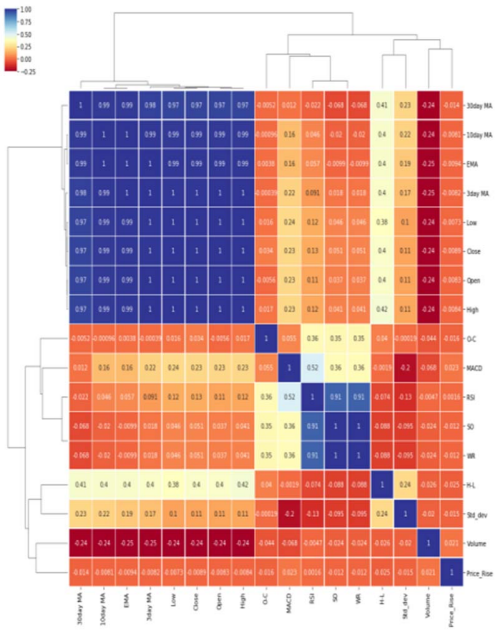
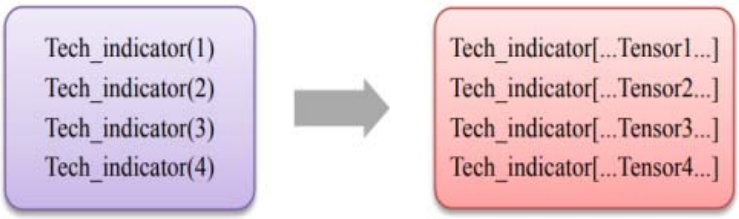


Fig. 5: Correlation Map for HDFC Stock

[Figure 1-3] Correlation Map for HDFC Stock. [2]

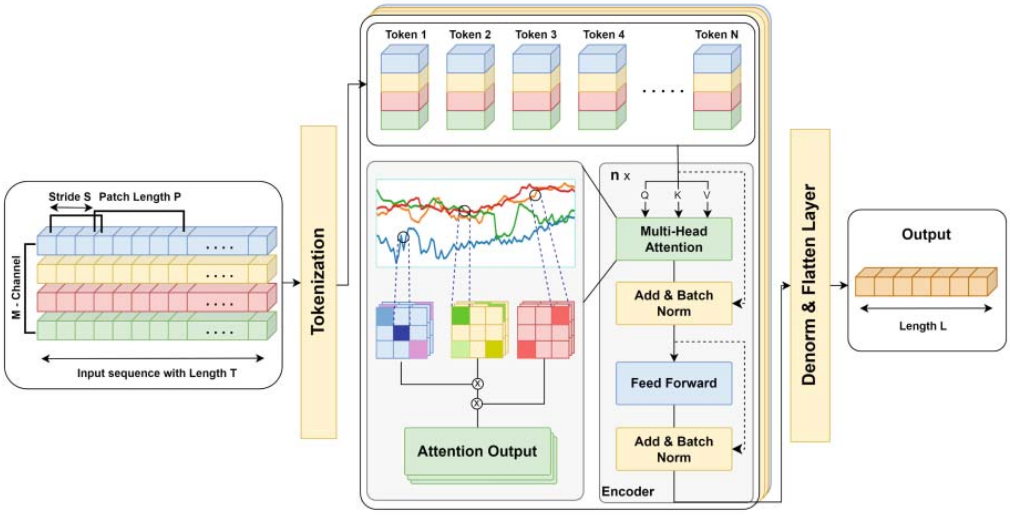
Next find the most effective technical tensor derived from this process [Figure 1-4] is then fed into the optimized LSTM model to generate the predicted price and corresponding buy and sell decisions.



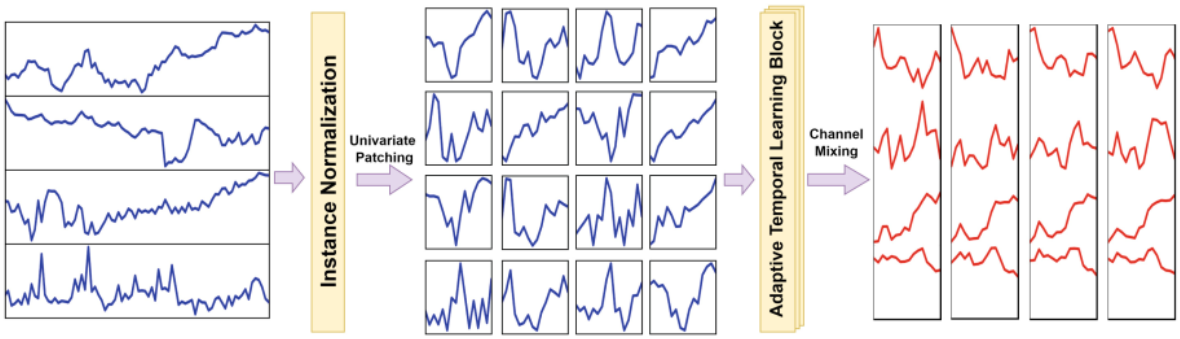
[Figure 1-4] Tensor Transformation Phenomenon [2]

2.2 Patched Channel Integration Encoder

Zhu, Zhuo hang, et al. [10] introduce tokenization as a preprocessing step before using the attention model for training. In their approach [Figure 2-1], the input data is divided into smaller data tokens and projected onto embedding vectors by [Figure 2-2] processing, which helps the model better understand the data's meaning and improves prediction accuracy.



[Figure 2-1] PCIE Model Overview. [10]



[Figure 2-2] Tokenization Process. [10]

2.2.1 Tokenization Process

The Tokenization Process [Figure 2-2] includes univariate patching, adaptive temporal learning, and channel mixing. In univariate patching, the input sequence is divided into patches of length P with stride S, which can be either overlapping or non-overlapping, as defined in Eq. (2-1).

$$N = \left|\frac{(L-P)}{S}\right| + 1 \qquad \text{Eq. (2-1)}$$

2.2.2 Adaptive Temporal Learning

The Adaptive Temporal Learning Block is designed to dynamically select the most effective temporal learning methodology from a range of options, including linear, series-independent linear and Multilayer-Perceptron (MLP) with the objective of producing learned vectors that best represent the underlying patterns for different datasets. as defined in Eq. (2-2) and Eq. (2-3).

$$X_{d\_patch}^{(i)} = W_{linear}^{(i)} * X_P^{(i)} + b_{linear}^{i} \qquad \text{Eq. (2-2)}$$

$$X_{d\_patch}^{(i)} = MLP(X_P^{(i)}) \qquad \text{Eq. (2-3)}$$

2.2.3 Channel Mixing

A channel mixing step will be performed on the output of the ATL block, a learnable additive position encoding $W_{pos} \in R^{d\_model\ X\ N}$ is applied to represent the order of the patches by Eq. (2-4).

$$X_{d\_model} = Flatten(X_{d_{patch}}^{(i)}, \dots, X_{d\_patch}^{(M)}) + W_{pos} \qquad \text{Eq. (2-4)}$$

The tensor is fed into the multi-head self-attention model to learn temporal vectors for the latent representation.

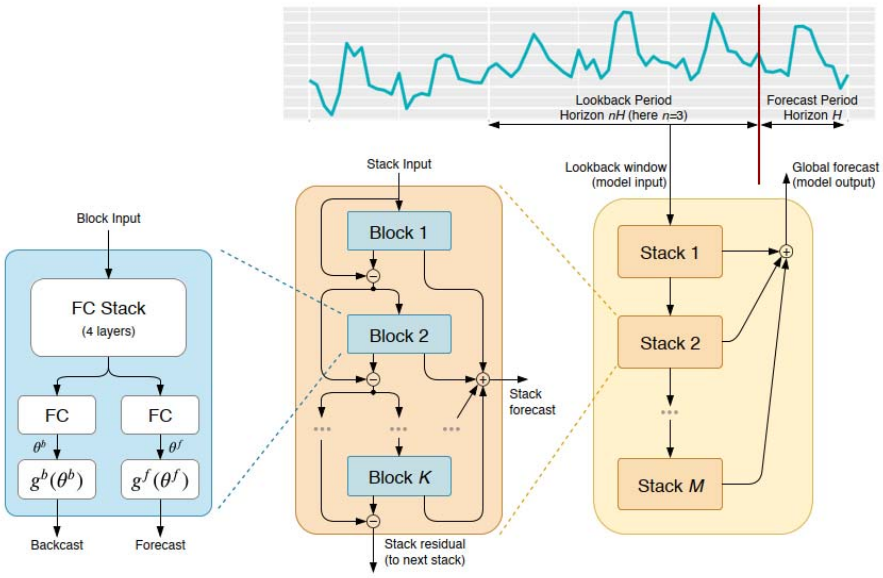
2.2.4 Feed-Forward Output Layers

And Feed-Forward Output Layers is used to obtain the multi-step stock price forecast with length Lf in a single step as Eq. (2-5), this is also called direct multi-step forecast.

$$X_{Lf} = W_f * Flatten(X_A) + b_{linear} \qquad \text{Eq. (2-5)}$$

2.3 Neural Basis Expansion Analysis for Interpretable Time Series Forecasting

Oreshkin, Boris N., et al. [12] propose a methodology called N-BEATS, which uses a fully connected network to stack empirical modes. [Figure 3-1]



[Figure 3-1] Proposed architecture [12]

2.3.1 Basic Building Block

Internally, the basic building block consists of two parts. The first part is a fully connected network that produces the forward θ_ℓ^f and the backward θ_ℓ^b predictors of expansion coefficients. The second part consists of the backward g_ℓ^b and the forward g_ℓ^f basis layers that accept the respective forward θ_ℓ^f and backward θ_ℓ^b expansion coefficients, project them internally on the set of basis functions and produce the backcast $\hat{x}_\ell$ and the forecast outputs $\hat{y}_\ell$ defined in the previous paragraph.

The operation of the first part of the ℓ-th block is described by Eq. (3-1) and Eq. (3-2):

$$h_{\ell,1} = FC_{\ell,1}(x_\ell)\,, \qquad h_{\ell,2} = FC_{\ell,2}(h_{\ell,1}), \qquad h_{\ell,3} = FC_{\ell,3}(h_{\ell,2}), \qquad h_{\ell,4} = FC_{\ell,4}(h_{\ell,3}) \quad \text{Eq. (3-1)}$$

$$\theta_\ell^b = LINEAR_\ell^b(h_{\ell,4})\,, \quad \theta_\ell^f = LINEAR_\ell^f(h_{\ell,4}) \quad \text{Eq. (3-2)}$$

LINEAR layer is simply a linear projection layer, i.e. $\theta_\ell^f = W_\ell^f(h_{\ell,4})$, The FC layer is a standard fully connected layer with RELU non-linearity, such that for $FC_{\ell,1}$ we have, for example:
$h_{\ell,1} = RELU(W_{\ell,1}x_\ell + b_{\ell,1})$

2.3.2 Outputs Via Basis Layers

The second part of the network maps expansion coefficients θ_ℓ^f and θ_ℓ^b to outputs via basis layers, by $\hat{y}_{\ell=}g_\ell^f(\theta_\ell^f)$ and $\hat{x}_{\ell=}g_\ell^b(\theta_\ell^b)$. Its operation is described by Eq. (3-3):

$$\hat{y}_{\ell=} \sum_{i=1}^{\dim(\theta_\ell^f)} \theta_{\ell,\mathrm{i}}^f v_\mathrm{i}^f \,, \hat{x}_{\ell=} \sum_{i=1}^{\dim(\theta_\ell^b)} \theta_{\ell,\mathrm{i}}^b v_\mathrm{i}^b \qquad \text{Eq. (3-3)}$$

Here v_i^f and v_i^b are forecast and backcast basis vectors, The function of g_ℓ^f and g_ℓ^b is to provide sufficiently rich sets $\{v_\mathrm{i}^f\}_{i=1}^{\dim(\theta_\ell^f)}$ and $\{v_\mathrm{i}^b\}_{i=1}^{\dim(\theta_\ell^b)}$ such that their respectiveoutputs can be represented

adequately via varying expansion coefficients θ_ℓ^f and θ_ℓ^b. As shown below, g_ℓ^b and g_ℓ^f can either be chosen to be learnable or can be set to specific functional forms to reflect certain problem-specific inductive biases in order to appropriately constrain the structure of outputs.

2.3.3 Architecture of Model

[Figure 3-1] (middle and right) The proposed architecture has two residual branches, one running over backcast prediction of each layer and the other one is running over the forecast branch of each layer. Its operation is described by Eq. (3-4):

$$x_\ell = x_{\ell-1} - \hat{x}_{\ell-1}, \hat{y} = \sum_\ell \hat{y}_\ell \qquad \text{Eq. (3-4)}$$

For all other blocks, the backcast residual branch x_ℓ can be thought of as running a sequential analysis of the input signal. The final forecast $\hat{y}$ is the sum of all partial forecasts.

Chapter 3: Hybrid model

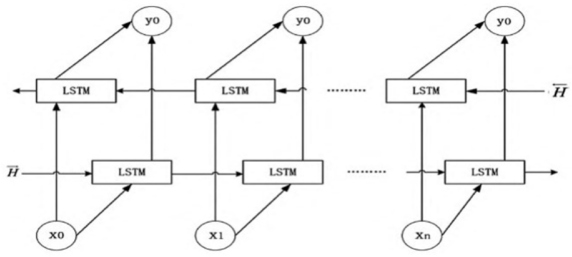
To achieve higher accuracy in prediction results, many researchers propose hybrid models to deal with segmented data by decomposing the time series into smaller parts or different elements, then applying suitable models to obtain precise predictions.

3.1 CNN-BiLSTM-Attention

Zheng, Haotian, et al. [1] propose the CNN-BiLSTM-Attention hybrid architecture. The CNN model is used to extract spatial features of time series data.

3.1.1 BiLSTM

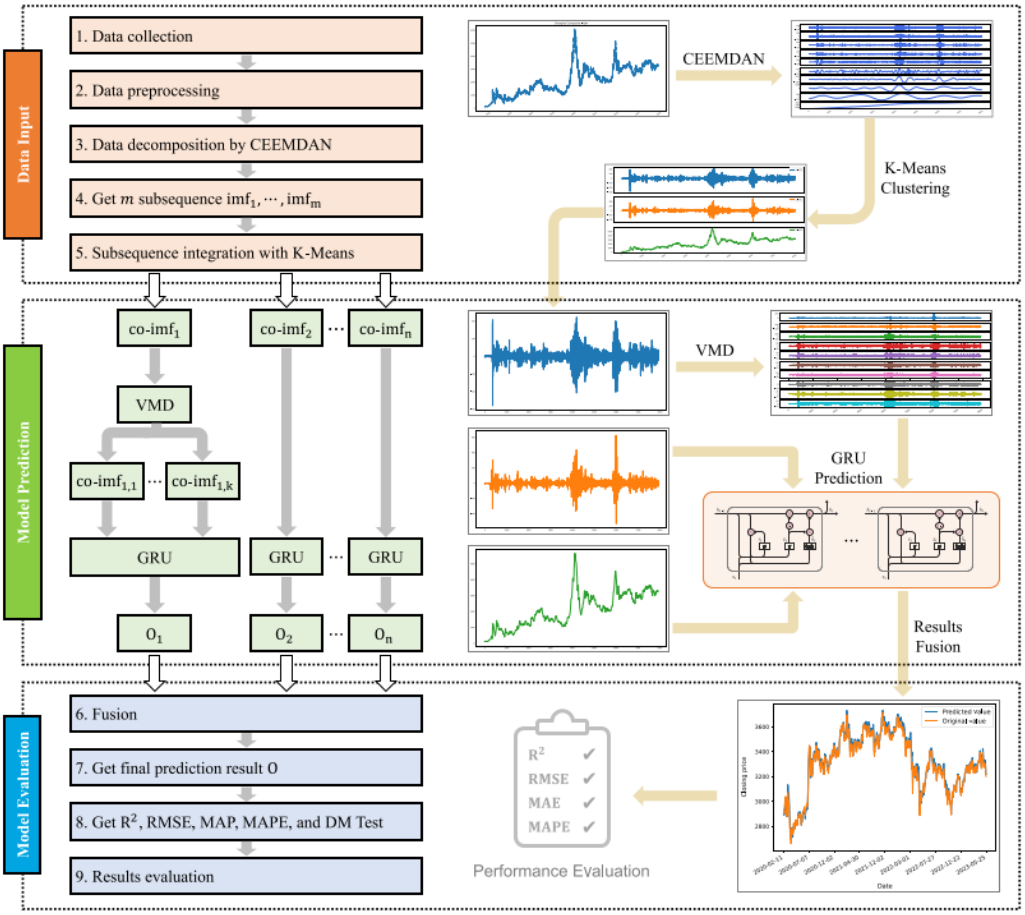
[Figure 4-1] Bi-directional Long Short-Term Memory (BiLSTM) is introduced for events that need to consider the impact of future data on historical data. The model can reference the influence of both historical and future data on the predicted results. Before training the model, The data is preprocessed first, including the processing of missing and duplicate values, and then normalized.



[Figure 4-1] BiLSTM expansion structure based on LSTM. [1]

3.2 Hierarchical Decomposition-based Forecasting Model

Y. Li, L. Chen, C. Sun, G. Liu, C. Chen and Y. Zhang [3] propose a HDFM model [Figure 5-1] using CEEMDAN to decompose data and then applying VMD to further decompose the highest frequency component. The processed data are then input into GRU, and different frequency data are weighted to achieve higher accuracy prediction.



[Figure 5-1] Overview of the proposed HDFM model.[3]

3.2.1 CEEMDAN

The CEEMDAN can be described as follows:

(1) Add white noise $\varepsilon_0\delta^k(t)$ to the original signal $x(t)$ to produce K different new series $x^k(t) = x(t) + \varepsilon_0\delta^k(t), k = 1,2,\dots,K, \delta^k(t)$ *is the white noise and* ε_0 *is the weight.* Then, the first IMF and the first residual can be obtained as Eq. (4-1) and Eq. (4-2)

$$\mathrm{IMF}_1(\mathrm{t}) = \frac{1}{\mathrm{K}}\sum_{\mathrm{k}=1}^{\mathrm{k}} \mathrm{IMF}_1^{\mathrm{k}}(\mathrm{t}) \qquad \text{Eq. (4-1)}$$

$$\mathrm{r}_1(\mathrm{t}) = \mathrm{x}(\mathrm{t}) - \mathrm{IMF}_1(\mathrm{t}) \qquad \text{Eq. (4-2)}$$

$\mathrm{IMF}_1^{\mathrm{k}}(\mathrm{t})$ represents the decomposed sub-series by EMD applied to the k-th new series $\mathrm{x}^{\mathrm{k}}(\mathrm{t})$.

(2) For n = 2, · · · , N, add white noise $\varepsilon_{n-1}E_{n-1}(\delta^k(t))$to the residual $r_{n-1}^k(t)$ to produce K different new residuals, $r_{n-1}^k(t) = r_{n-1}(t) + \varepsilon_{n-1}E_{n-1}(\delta^k(t))$, k = 1, 2, · · · , K. E_{n-1} (·) is defined as the (n − 1)-th IMF of a signal produced by EMD. Then, the n-th IMF and the n-th residual are obtained as Eq. (4-3) and Eq. (4-4):

$$\mathrm{IMF}_{\mathrm{n}}(\mathrm{t}) = \frac{1}{\mathrm{K}}\sum_{\mathrm{k}=1}^{\mathrm{k}} E_1\left(r_{n-1}^k(\mathrm{t})\right), \mathrm{n} = 2,\dots,\mathrm{N} \qquad \text{Eq. (4-3)}$$

$$\mathrm{r_n(t)} = r_{n-1}(\mathrm{t}) - \mathrm{IMF_n(t)}, \mathrm{n} = 2, \ldots, \mathrm{N} \qquad \text{Eq. (4-4)}$$

(3) Step 2 is repeated until the obtained residual can no longer be decomposed. The final residual of CEEMDAN is obtained as Eq. (4-5):

$$\mathrm{R(t)} = \mathrm{x(t)} - \sum_{\mathrm{n=1}}^{\mathrm{N}} IMF_n(t) \qquad \text{Eq. (4-5)}$$

3.2.2 VMD

The VMD framework consists of two stages: construction and solving. In the construction stage, for a given signal f , the variational problem is solved by minimizing the sum of the estimated bandwidth for each mode u_k (k = 1, 2, · · · , K). In reverse, the modes collectively reproduce the signal f . The solving problem can be formulated as the following constrained optimization problem Eq. (4-6):

$$\begin{aligned} &\min_{\{u_k\},\{\omega_k\}} \left\{ \sum_{k=1}^{K} \left\| \partial_t \left[\left(\delta(t) + \frac{j}{\pi t} \right) \otimes u_k(t) \right] e^{-j\omega_k t} \right\|_2^2 \right\} \\ &\text{s.t.} \quad \sum_{k=1}^{K} u_k = f \end{aligned} \qquad \text{Eq. (4-6)}$$

where $\{u_k\} := \{u_1, u_2, \cdots, u_k\}$ represents the set of all modes, and $\{\omega_k\} := \{\omega_1, \omega_2, \cdots, \omega_k\}$ represents the center frequencies of each mode.$\delta(t)$ is the Dirac distribution.

In the solving stage, the above constrained optimization problem is transformed into an unconstrained one by introducing the secondary penalty factor α and the Lagrange multiplier λ as follows Eq. (4-7):

$$\begin{aligned} &\mathcal{L}(\{u_k\}, \{\omega_k\}, \lambda) \\ &= \alpha \sum_{k=1}^{K} \left\| \partial_t \left[\left(\delta(t) + \frac{j}{\pi t} \right) \otimes u_k(t) \right] e^{-j\omega_k t} \right\|_2^2 \\ &+ \left\| f(t) - \sum_{k=1}^{K} u_k(t) \right\|_2^2 + \left\langle \lambda(t), s(t) - \sum_{k=1}^{K} u_k(t) \right\rangle \end{aligned} \qquad \text{Eq. (4-7)}$$

Eq. (4-7) is solved by using the ADMM strategy, and the optimal solution of u_k , $\omega_k\}$ and λ are obtained as follows Eq. (4-8):

$$\hat{u}_k^{n+1}(\omega) = \frac{\hat{f}(\omega) - \sum_{i \neq k} a_i(\omega) + \frac{\hat{\lambda}(\omega)}{2}}{1 + 2\alpha\,(\omega - \omega_k)^2}$$

$$\hat{\omega}_k^{n+1}(\omega) = \frac{\int_0^\infty \omega \left|\hat{u}_k(\omega)\right|^2 d\omega}{\int_0^\infty \left|\hat{u}_k(\omega)\right|^2 d\omega}$$

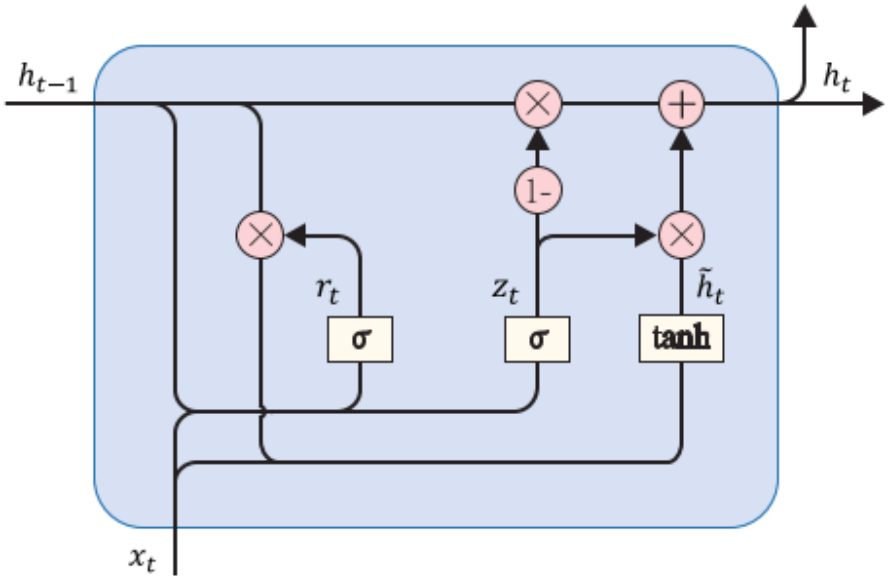
$$\hat{\lambda}^{n+1}(\omega) = \hat{\lambda}^n(\omega) + \tau\left(\hat{f}(\omega) - \sum_{k=1}^{K} \hat{u}_k^{n+1}(\omega)\right)$$

Eq. (4-8)

where $\hat{u}_k^{n+1}(\omega)$, $\hat{u}_i(\omega)$, $\hat{f}(\omega)$, and $\lambda(\omega)$ denote the Fourier transform of $u_k^{n+1}(t)$, $u_i(t)$, $f(t)$, and $\lambda(t)$, respectively. τ is the tolerance of noise.

3.2.3 GRU

The GRU memory cell consists of only two parts as shown in [Figure 5-2]: the update and reset gates. In [Figure 5-2]: z_t denotes the update gate, r_t represents the reset gate, and $h_{t-1}$and h_t denote the previous and current hidden states, respectively. $\tilde{h}_t$ is the candidate hidden state.



[Figure 5-2] The internal structure of a GRU unit. [3]

The update gate z_t Eq. (4-9) determines how much of the previous hidden state h_{t-1}need to be retained, and how much will be replaced by the new information in $\tilde{h}_t$.

$$z_t = \sigma(w_z[h_{t-1}, x_t] + b_z) \qquad \text{Eq. (4-9)}$$

The reset gate r_t Eq. (4-10) determines how much of the previous hidden state h_{t-1}will be discarded, and how much will be used to calculate $\tilde{h}_t$.

$$r_t = \sigma(w_r[h_{t-1}, x_t] + b_r) \qquad \text{Eq. (4-10)}$$

This step primarily computes the candidate hidden state $\tilde{h}_t$ that is to be mixed with the previous hidden state h_{t-1} through the update gate z_t as Eq. (4-11).

$$\tilde{h}_t = tanh(w_h[r_t \cdot h_{t-1}, x_t] + b_h) \qquad \text{Eq. (4-11)}$$

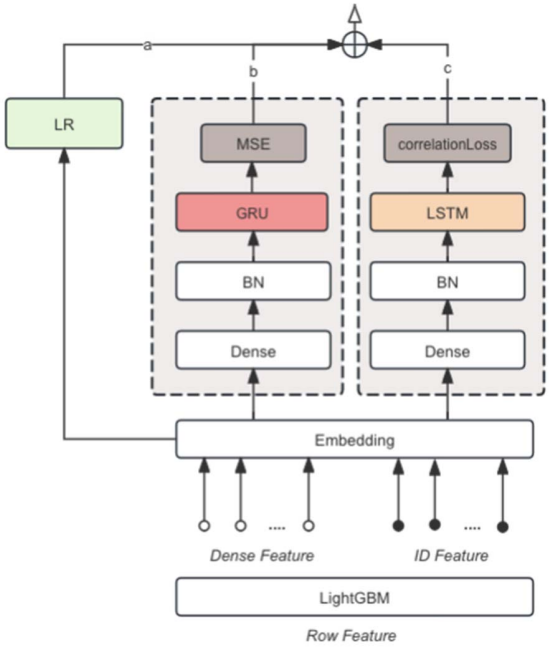
The final step is to calculate the current hidden state h_t with a linear combination of the previous hidden state h_{t-1} and the candidate hidden state $\tilde{h}_t$, weighted by the update gate z_t as Eq. (4-12).

$$\tilde{h}_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t \qquad \text{Eq. (4-12)}$$

3.3 Ensemble Approach to Stock Price Prediction

M. Sui, C. Zhang, L. Zhou, S. Liao and C. Wei [4] combine LightGBM, LSTM, and GRU layers with a hyperparameter tuning mechanism to achieve high accuracy results as shown in [Figure 6-1]

Dense Layers [DN] is to Introduce non-linearity and reduces dimensionality.
Batch Normalization [BN] is use to standardize the inputs.



[Figure 6-1] Model Ensemble Pipeline. [4]

The Logistic Regression (LR) is used as a robust baseline model, which is integrated with the deep learning models to enhance stability and reduce overfitting. The logistic regression model is defined as Eq. (4-5):

$$\hat{y} = \sigma(W \cdot X + b) \qquad \text{Eq. (4-5)}$$

σ is the sigmoid function, This model helps in leveraging linear relationships and serves as a comparative baseline to complex models.

3.3.1 LightGBM

LightGBM is a gradient boosting framework that uses tree- based learning algorithms. It is designed for high efficiency and scalability. The LightGBM model uses the following key steps: Feature Engineering: Creation of new features such as returns, moving averages, and volatility. Tree-based Learning: Utilizes decision trees to iteratively improve model performance.

The objective function optimized by LightGBM as Eq. (4-6):

$$L = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{j=1}^{m} \Omega(f_j) \qquad \text{Eq. (4-6)}$$

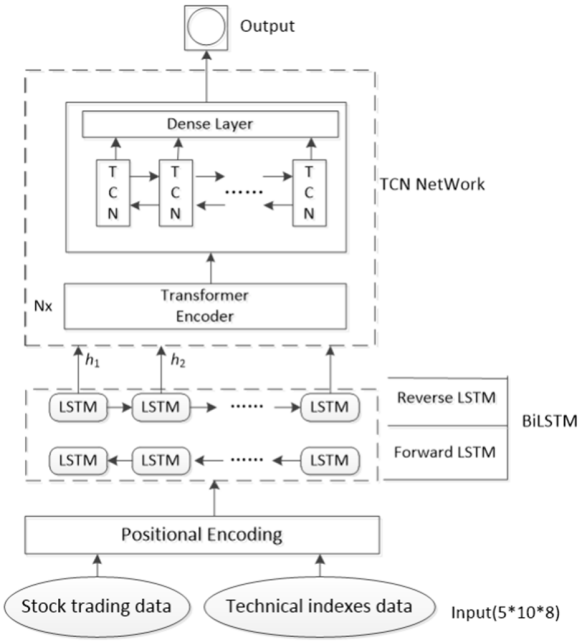
The model is trained using multiple loss functions such as MSE Eq. (4-7) and Correlation Loss Eq. (4-8) to enhance its predictive capabilities:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad \text{Eq. (4-7)}$$

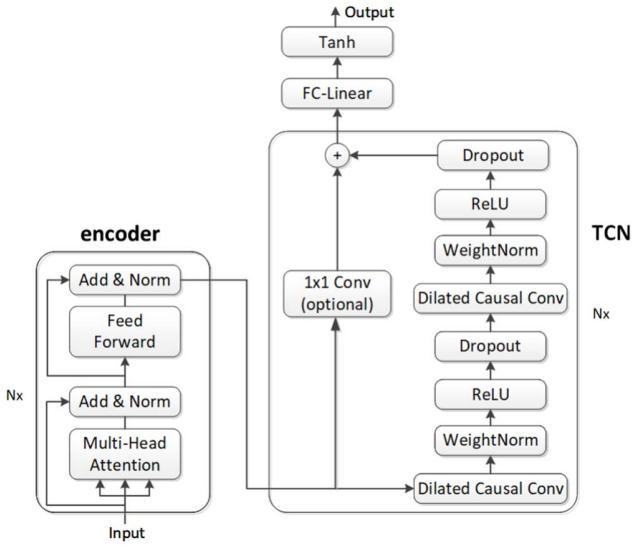
$$Correlation\ Loss = 1 - \frac{Cov(y,\hat{y})}{\sigma_y \sigma_{\hat{y}}} \qquad \text{Eq. (4-8)}$$

3.4 BiLSTM-MTRAN-TCN

S. Wang [5] proposes the BiLSTM-MTRAN-TCN model as [Figure 7-1], which replaces the original decoder part with a TCN layer and a fully connected layer as [Figure 7-2]. This model shows high accuracy and generalization ability, while avoiding timeliness issues.



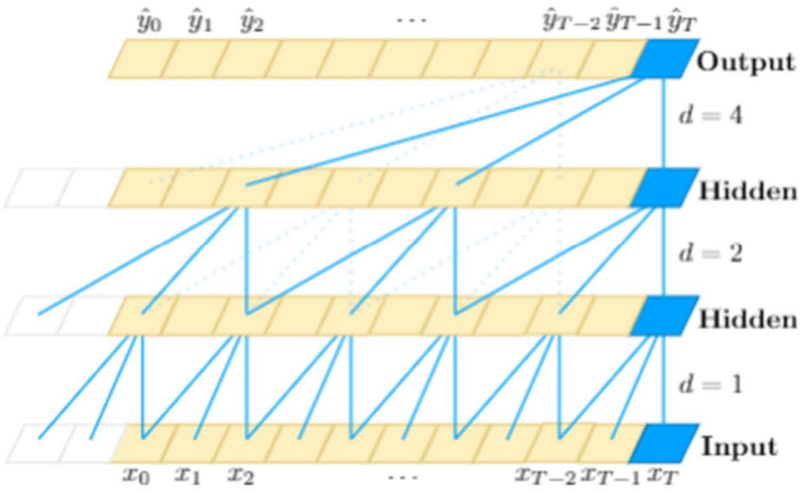
[Figure 7-1] BiLSTM-MTRAN-TCN model structure diagram. [5]



[Figure 7-2] Time convolutional neural network (TCN). [5]

3.4.1 TCN

The network structure of TCN mainly consists of causal convolution, dilation convolution and residual connections. Each convolution layer is a causal convolution with a unidirectional structure, as shown in [Figure 7-3].



[Figure 7-3] Modified overall structure of transformer (MTRAN-TCN).
Modified transformer, abbreviated as MTRAN-TCN. [5]

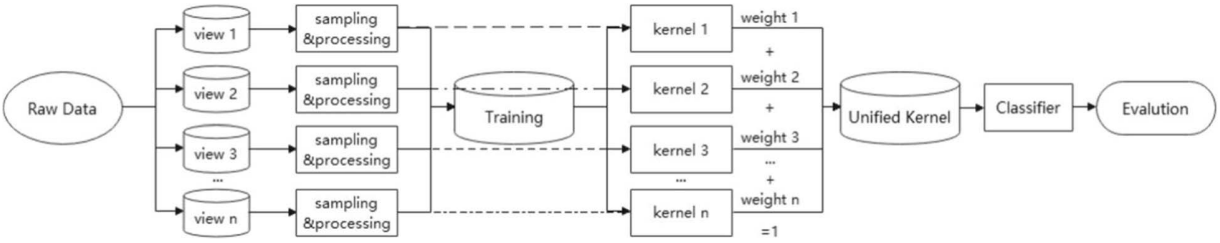
Causal convolution ensures that the output at time T is only convolved with elements that occurred before time T, which avoids future information leakage. And it can accept input sequences of any length and output the same length. The calculation of causal convolution, as shown in Eq. (5-1)

$$F(s) = \sum_{i=0}^{k-1} f(i)\, xs - di \qquad \text{Eq. (5-1)}$$

The dilation convolution is used to capture longer dependency information without stacking more layers or adding pooling layers to obtain larger a receptive field.

3.5 MVL-SVM

Long, Wen, et al. [6] propose MVL-SVM algorithm combines a support vector machine and multi-view heterogeneous data [Figure 8-1] to predict stock price movements more accurately.



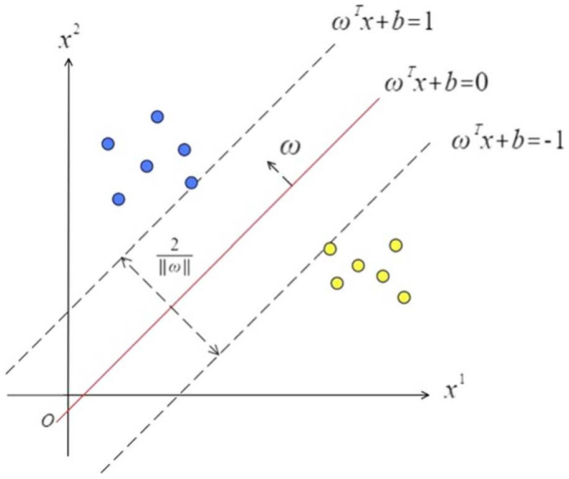
[Figure 8-1] Sketch map of multiple kernel learning. [6]

3.5.1 SVM

The Support vector machine (SVM) has been widely applied to solve classification problems owing to its performance. It can learn from a set of two-class training instances and divide new instances into one of the classes to solve classification problems. The main idea of linear SVM is to find a " maximum margin hyperplane," defined as $g(x) = \omega^T x + b$ so that the two classes of samples can be accurately classified by this hyperplane and the sum of distances between the hyperplane and the closest point of each class is maximized. Mathematically, the classification problem is equivalent to solving the minimization problem as follows Eq. (6-1):

$$min \frac{1}{2}\omega^T \omega + C \sum_{i=1}^{n} \xi_i \qquad \text{Eq. (6-1)}$$

$$s.t.\ yi(\omega^T x_i + b) + \xi_i \geq 1, \forall 1 \leq i \leq n,\ \xi_i \geq 0, \forall 1 \leq i \leq n.$$



[Figure 8-2] A 2-dimensional classification instance using SVM. [6]

A 2-dimensional example is shown in [Figure 8-2] to clearly demonstrate the workings of the linear SVM. Here, samples of different colors come from different classes. The red line represents the maximum margin hyperplane obtained by training the samples.

3.5.2 Multi-View Learning

Multi-view learning by selecting the appropriate kernels and kernel combination for training, each data source can be trained with the corresponding optimal kernel function; therefore, the model can perform better than the single-kernel model, the weighted summation kernel are used in this study, and the kernel function can be written as Eq. (6-2)

$$K(x_i, x_j) = \sum_m \beta_m k_m(x_i, x_j), \beta_m \geq 0, \sum_m \beta_m = 1 \qquad \text{Eq. (6-2)}$$

$k_m(x_i, x_j)$ denotes the m-th kernel, β_m represents the weight of kernel $k_m(x_i, x_j)$.

Here, we build an MVL-SVM model to predict stock price movement using one-day market data and financial news. The news combines length normalization ("l"), term frequency ("t") and collection frequency ("c") to calculate the weights of words. By normalizing the weight of words, the influence of article length can be avoided, and the importance of word frequency is weakened to a certain extent. This represents news articles in the following form:

$$news_t = (w_{t,1}, w_{t,2}, \dots, w_{t,m})$$

where $news_t$ represents the news vector on day_t and $w_{t,m}$ is the weight of wordm in $news_t$, which is expressed as Eq. (6-3):

$$w_{t,m} = \frac{(\log(f_{t,m})+1.0)*log\frac{1}{Fm})^2}{\sum_{j=1}^{M}[(\log(f_{t,j})+1.0)*log\frac{1}{Fj}]^2} \text{ Eq. (6-3)}$$

where $f_{t,m}$ represents the occurrence frequency of wordm in $news_t$ (term frequency), Fm is the occurrence frequency of $Wordm$ in the news corpus (collection frequency).

Because the news corpus involves many words, but only a small portion of words is contained in each news item, we use χ^2 statistics Eq. (6-4) to extract features of the text. Instead of using all the words in the news corpus, it selects words that contribute more to text classification to make computation easier and prevent overfitting. For word w and category c, we define A as the number of times w and c co-occur, B as the number of times w occurs without c, C as the number of times c occurs without w, D as the number of times neither c nor w occurs, and n is the sample size.

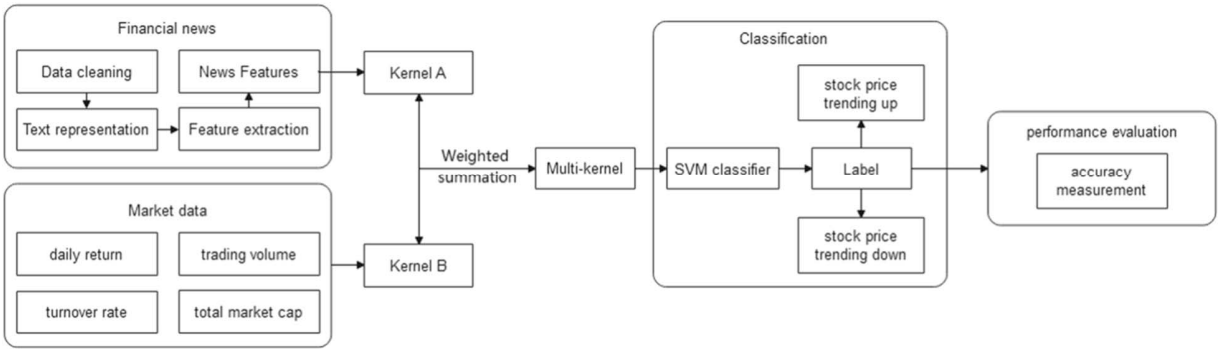
$$\chi^2(w,c) = \frac{n \times (AD-BC)^2}{(A+B)\times(A+C)\times(B+D)\times(C+D)} \quad \text{Eq. (6-4)}$$

Words with higherχ^2 scores are considered more informative for prediction, so we use χ^2 scores to select the optimal number of words with the best prediction performance as the dimension of news.

3.5.3 Model Framework

By applying [Figure 8-3] framework, the high-dimensional news vector obtained from the Chinese news text processing methods and the four-market data were all input to the MVL-SVM algorithm.

After training on the training set, the algorithm can choose the optimal kernel for each data source and the optimal kernel combination weights. Therefore, combining multiple kernels can successfully fuse the multi-view data, and the new kernel can be input into the SVM classifier for classification.



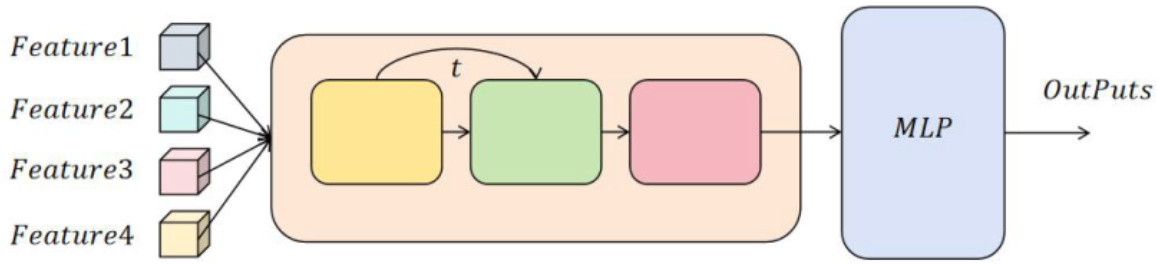
[Figure 8-3] Multi-view learning framework of stock price prediction model. [6]

The input matrix of market data MD and News is formulated in Eq. (6-5) where each row of MD and News denotes a vector, and the labels of these vectors are also shown in Eq. (6-5)

$$MD = \begin{bmatrix} r1 & tv_1 & tr_1 & mc_1 \\ r2 & tv_2 & tr_2 & mc_2 \\ . & . & . & . \\ . & . & . & . \\ r_{n-2} & tv_{n-2} & tr_{n-2} & mc_{n-2} \\ r_{n-1} & tv_{n-1} & tr_{n-1} & mc_{n-1} \end{bmatrix}, news = \begin{bmatrix} news_2 \\ news_3 \\ . \\ . \\ news_{n-1} \\ news_n \end{bmatrix}, Label = \begin{bmatrix} tag_2 \\ tag_3 \\ . \\ . \\ tag_{n-1} \\ tag_n \end{bmatrix} \quad \text{Eq. (6-5)}$$

3.6 Capturing Temporal Dependencies and Multi-Dimensional Features

Yao, Y. [7] improves the Transformer model by combining feature fusion, self-attention, noise filtering in the input data, and an adaptive time window with a weight adjustment mechanism to achieve better results as shown in [Figure 9-1]



[Figure 9-1] Model overall framework diagram. [7]

In model design, first performed feature fusion on the input data, combining the fundamental data of the stock market (such as financial report data, macroeconomic indicators, etc.), technical indicators (such as moving averages, relative strength index, etc.) and historical stock price data.

3.6.1 Adaptive Time Window

Second, introduced adaptive time window technology, adjusting the size of the time window according to the dynamic characteristics of historical data, to model stock market fluctuations on different time scales. Specifically, a time window size is set to t_W ,which is adaptively adjusted at each time to t, depending on the changes in historical stock prices and other auxiliary features.

3.6.2 Weight adjustment

Next, we introduced a weight adjustment mechanism Eq. (7-1) in the Transformer's self-attention mechanism to improve the model's sensitivity to key features.

$$S_{match} = \sum_{i=1}^{n} W_i \cdot Attention(x_i, x_t) \qquad \text{Eq. (7-1)}$$

Among them, S_{match} represents the matching score, x_i represents the input of feature i, x_t is the stock price feature at the current moment, W_i is the weight of the feature, $Attention(x_i, x_t)$ is the attention value calculated by the self-attention mechanism, and n is the total number of features.

In addition, we used wavelet transform and adaptive filtering technology to remove high-frequency noise in the data, first perform a wavelet transform on the stock market data to decompose the data into components of different frequencies. Then, we use an adaptive filtering method to remove the high-frequency noise part and retain the low-frequency components, thereby improving the quality of the data.

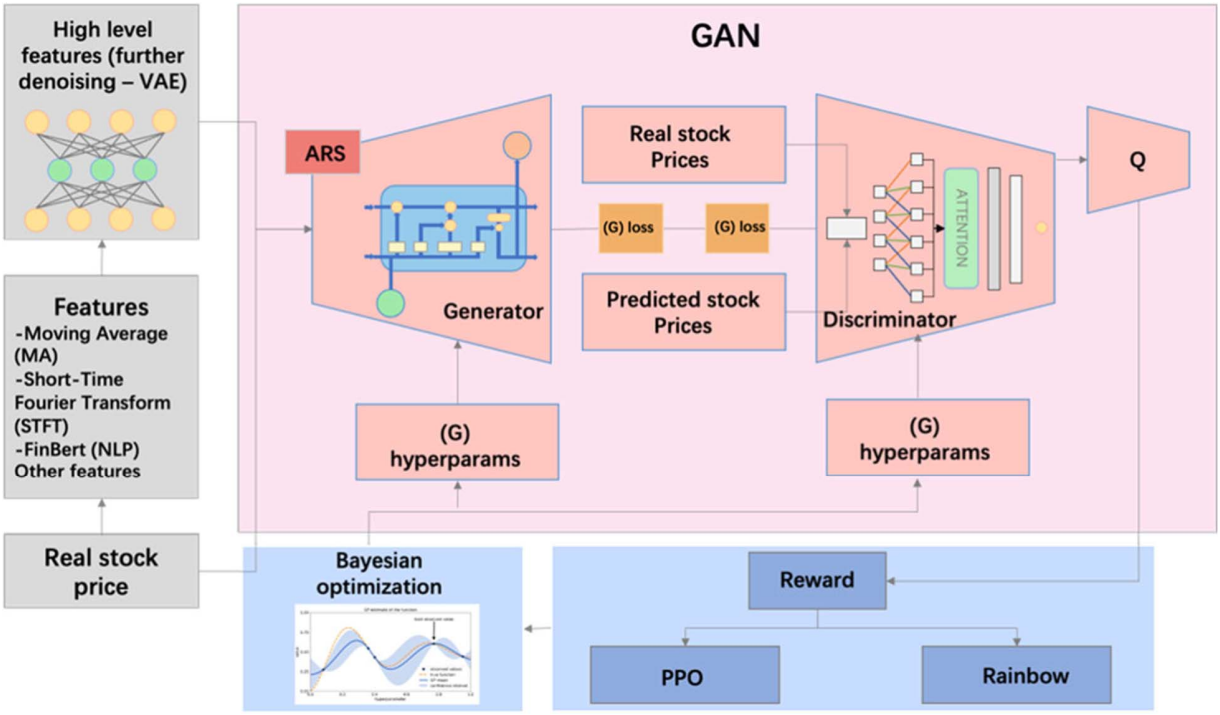
During the optimization process, the loss function of the model is optimized by combining the mean square error (MSE) and the weighted attention loss. Specifically, the loss function L can be expressed as Eq. (7-2):

$$L = \lambda_1 \cdot MSE(y_{pred}, y_{true}) + \lambda_2 \cdot Loss_{Attention}(S_{match}) \qquad \text{Eq. (7-2)}$$

y_{pred} and y_{true} represent the predicted value and true value of the model respectively, $MSE(y_{pred}, y_{true})$ is the mean square error loss, $Loss_{Attention}$ is the loss related to the self-attention mechanism, and λ_1 and λ_2 are hyperparameters used to balance the weights of the two losses.

3.7 GANs and transformer-based attention mechanisms

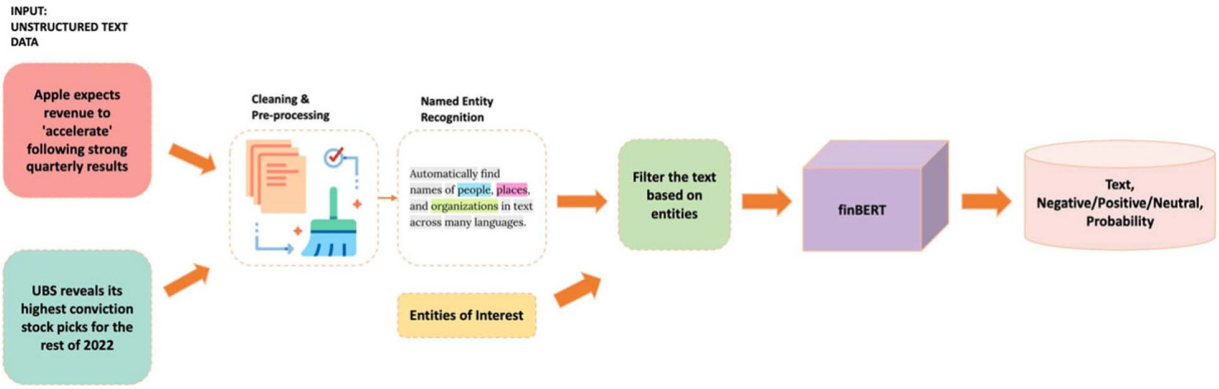
Li, S., Xu, S [8]. Propose a Model architecture as [Figure 10-1] that apply the adaptive impact of daily news technical indicators and Fourier components into a VAE, then use the important features to train GANs to generate accurate results.



[Figure 10-1] Overview of the model architecture. This figure illustrates the model architecture employed in our stock price prediction framework, which incorporates three key components: data preprocessing, generative adversarial networks (GANs), and hyperparameter tuning. [8]

3.7.1 FinBERT

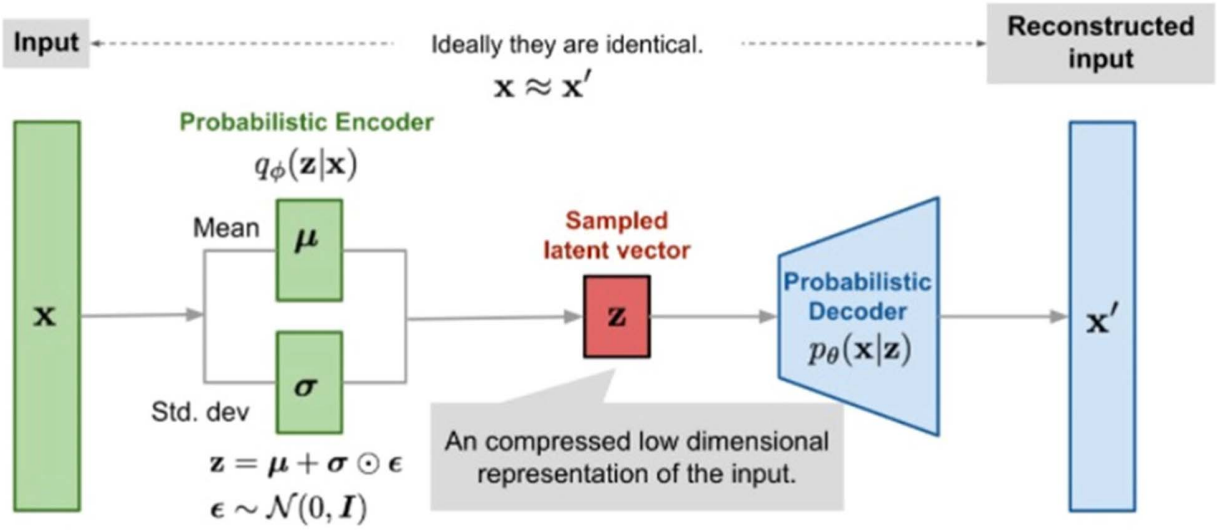
FinBERT [Figure 10-2], a specific model based on BERT (bidirectional encoder representations from transformers), is a useful tool for analyzing financial mood. This model is specifically designed to understand and evaluate the sentiment and contextual information contained in financial text data, such as news stories, earnings reports, and social media posts. Its major goal is to extract insights relevant to financial decision-making processes.



[Figure 10-2] Workflow of FinBERT. This figure visually depicts the workflow of our model and showcases the functionality of FinBERT in relation to our model and the social media news data. [8]

3.7.2 VAE

The variational autoencoder (VAE) [Figure 10-3] is a generative model that combines autoencoder with variational inference techniques. Its primary goal is to learn a latent representation of input data by encoding it into a lower-dimensional latent space and then decoding it to reconstruct the original data



[Figure 10-3] Workflow of VAE. This figure illustrates the key components of the variational autoencoder (VAE) architecture, namely the encoder, latent space, and decoder. Shubham P [13]

VAE it is made up of an encoder and a decoder network, the encoder takes input data and maps it to a latent space using mean (μ) and variance (σ^2) vectors, which parameterize a multi-variate Gaussian distribution such as $P(\mathrm{X}|z;\theta)$. The decoder network reconstructs the original data from samples in the latent space. The VAE is trained by reducing the evidence lower bound (ELBO), which contains the reconstruction error term and a latent space distribution regularization term.

The ELBO (Eq. 8-1) is calculated by taking the reconstructed data's probability given the latent variable and dividing it by the Kullback–Leibler divergence between the approximate posterior and the prior distribution, Monte Carlo sampling and backpropagation are used for optimization.

$$\ln \mathrm{p}(x) = ln \int_z \mathrm{p}(x,z) = ln \int_z \mathrm{p}(x,z) \frac{q(z|x)}{q(z|x)} \geq \mathbb{E}_q(z|x) \left[ln \frac{\mathrm{p}(x,z)}{q(z|x)} \right]$$

$$= \mathbb{E}_q(z|x) \left[ln \frac{\mathrm{p}(x|z)\mathrm{p}(z)}{q(z|x)} \right] = \mathbb{E}_{q(z|x)}[\ln p(x|z)] + \mathbb{E}_{q(z|x)} \left[ln \frac{\mathrm{p}(z)}{q(z|x)} \right]$$

$$= \mathbb{E}_{q(z|x)}[\ln p(x|z)] + \int_z q(z|x) ln \frac{\mathrm{p}(z)}{q(z|x)} = \mathbb{E}_{q(z|x)}[\ln p(x|z)] - D_{KL}[q(z|x) \,||\, \mathrm{p}(z)]$$

$$= likelihood \; - \; KL \; ; KL(P \,||\, Q) = \sum p_i(x) ln \frac{\mathrm{p}_i(x)}{q_i(x)} \qquad \text{(Eq. 8-1)}$$

3.7.3 GANs

Generative adversarial networks (GANs) are a category of deep learning models including a pair of neural networks, namely a Generator and a Discriminator. The Generator algorithm acquires knowledge of the fundamental data distribution and produces synthetic data samples that exhibit similarities to the authentic data. In contrast, the Discriminator's objective is to differentiate between authentic and artificially generated samples. The training process involves the simultaneous training of two components, namely the Generator and the Discriminator, in an adversarial manner. The primary objective of the Generator is to develop synthetic data that closely

resembles real data, with the intention of deceiving the Discriminator. Conversely, the Discriminator aims to accurately distinguish between genuine and synthetic samples. The objective function (Eq. 8-2) of GAN can be mathematically formulated as follows:

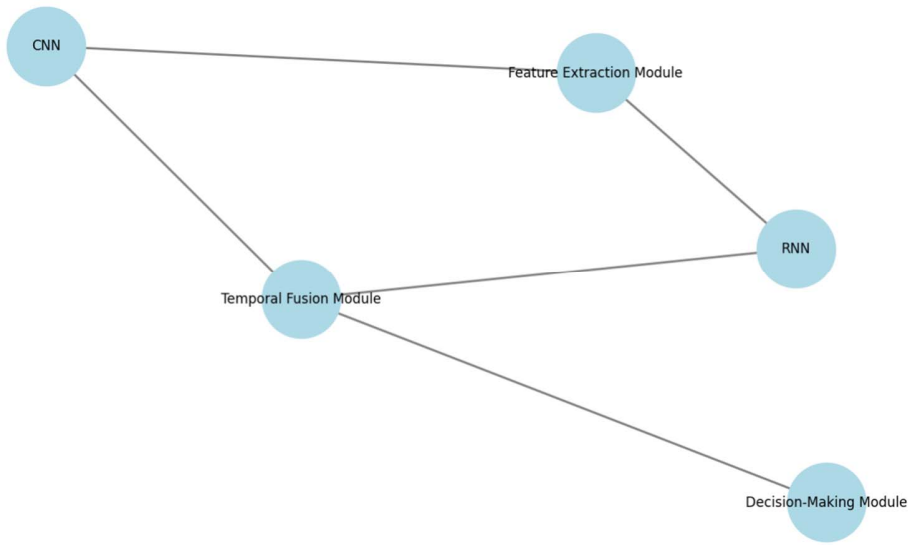
$$\min_G \max_D V(D,G) = \mathbb{E}_{Xreal \sim Preal}[logD(Xreal)] \mathrel{+}= \mathbb{E}_{z \sim Pz}[log(X1 - D(G(z)))] \qquad \text{(Eq. 8-2)}$$

$V(D,G)$ represents the value function of the GAN, $Preal$ is the real data distribution, Pz is the noise distribution, $Xreal$ is a real sample, $G(z)$is a generated sample, $D(Xreal)$ is the discriminator's output for a real sample, and $D(G(z))$ is the discriminator's output for a generated sample.

Hyperparameter tuning leverage probabilistic modeling, handle limited evaluations, search for global optima. The objective function of Bayesian optimization is root mean squared error (RMSE). After the evaluation, the algorithm will update its surrogate model which will capture the relationship between the hyper-parameters and the objective function.

3.8 High-Frequency Trading Strategies

Cao, G., Zhang, Y., Lou, Q., & Wang, G. [9] focus on high-frequency trading. They use an adaptive CNN+RNN structure to extract features from stock prices, combine them with an attention mechanism, and then input into a Markov Decision Process (MDP) framework to make active decisions for a higher win rate, the model architecture as shown in [Figure 11-1]



[Figure 11-1] Multi-Time Scale DRL Framework for HFT Optimization. [9]

3.8.1 Markov Decision Process

The high-speed trading problem (HFT) optimization is designed as a Markov Decision Process (MDP) to leverage the power of deep learning (DRL). State Space S represents the market and the agent's position, state space A includes the business decisions, and the reward space R includes profit/loss and exchange costs.

The state St at time t is defined as a vector containing relevant market features:

$$St = [pt, vt, dt, ot, ht, lt, ct, it]$$

Action space A consists of three possible actions: buy, sell, or hold. The reward function Rt as Eq. (9-1) is designed to maximize the Sharpe ratio while penalizing excessive trading:

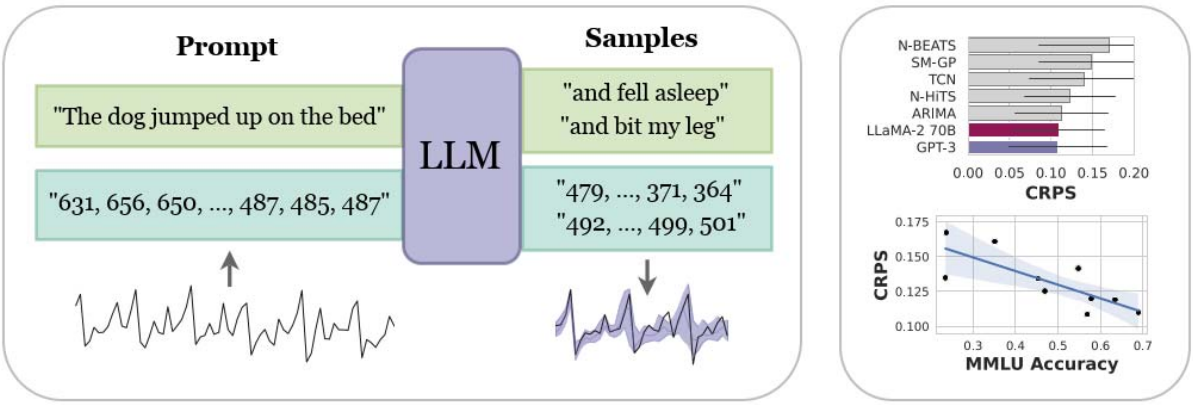
$$Rt = \frac{Pt - PT - 1}{\sigma t - \lambda |at|} \quad \text{Eq. (9-1)}$$

Pt is the portfolio value at time t, σt is the portfolio volatility, it is the action taken, and λ is a transaction cost parameter.

The feature extraction module employs convolutional neural networks (CNNs) to process high-frequency order book data and recurrent neural networks (RNNs) for lower-frequency price and volume data.

3.9 Large language models are zero-shot time series forecasters

Gruver, Nate, et al. [11] utilize existing LLM models to predict data without training any additional models as [Figure 12-1] shown.



[Figure 12-1] LLMTIME, a method for time series forecasting with large language models (LLMs) by encoding numbers as text and sampling possible extrapolations as text completions. LLMTIME can outperform many popular time series methods without any training on the target dataset (i.e. zero-shot). The performance of LLMTIME also scales with the power of the underlying base model. [11]

3.9.1 Language models for time series

Language modeling Language models are trained on a collection of sequences, $\mathcal{U} = \{U_1, U_2, \dots U_i, \dots, U_N\}$, where $U_i = \{u_1, u_2, \dots u_j, \dots, u_{ni}\}$ and each token, u_i, belongs to a vocabulary $\mathcal{V}$. Large language models typically encode an autoregressive distribution, in which the probability of each token is only dependent on the previous tokens in the sequence, $p_\theta(U_i) = \prod_{j=1}^{n_i} p_\theta(u_j | u_{0:j-1})$. The parameters, θ, are learned by maximizing the probability of the entire dataset $p_\theta(\mathcal{U}) = \prod_{i=1}^{N} p_\theta(U_i)$.

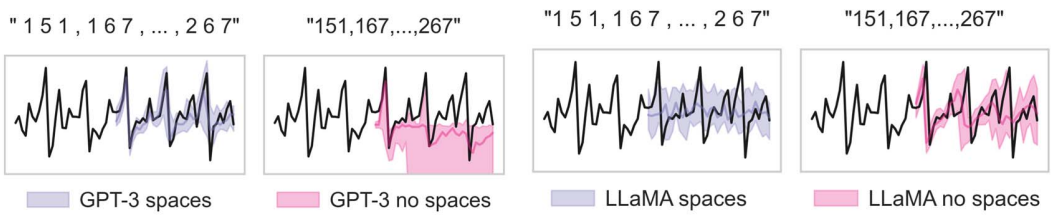
Time series data typically takes the exact same form as language modeling data, as a collection of sequences $\{U_i = \{(, u_2, \dots u_j, \dots, u_{ni})\}$ but in time series u_j is numerical. Because language models are built to represent complex probability distributions over sequences, they are theoretically well-suited for time series modeling.

The other challenge of applying language models to time series data is proper evaluation. Continuous ranked probability score (CRPS) captures distributional qualities and can compare models that generate samples without likelihoods. For a single prediction, the CRPS score is defined against the estimated cumulative distribution function (CDF), $\hat{F}$ as $CRPS(\hat{F}, \mathrm{y}) = \int_{\mathbb{R}} (\hat{F}(z) - \mathbb{I}_{(z-y)>0})^2 dz$. where $\hat{F}(z)$ is the empirical CDF produced by sampling forecasts and $\mathbb{I}$ is the indicator function. While CRPS is an improvement over MAE, it also ignores key structures in the data, such as correlations between time steps. Fortunately, language models can assign likelihoods to full sequences of time series data, and we show how a small modification to an LLM's discrete likelihood can yield a continuous density that is useful for model comparison.

3.9.2 Tokenization

Tokenization is particularly important because it directly influences how patterns form within tokenized sequences and the types of operations that language models can learn. Separate the digits with spaces to force a separate tokenization of each digit and use a comma (",") to separate each time step in a time series. Because decimal points are redundant given a fixed precision, we drop them in the encoding to save on

context length. Thus, with e.g. 2 digits of precision, we pre-process a time series as follows before feeding into the tokenizer:



[Figure 12-2] Careful tokenization is important for good forecasting with LLMs. Using the Australian Wine dataset from Darts, with values [151, 167, ..., 267], we show the tokenization used by GPT-3 and LLaMA-2 and the corresponding effect on forecasting performance. Added spaces allow GPT-3 to create one token per digit, leading to good performance. LLaMA-2, on the other hand, tokenizes digits individually, and adding spaces hurts performance. [11]

In [Figure 12-2], show that the added spaces of this encoding are helpful for GPT models, preventing the model from getting derailed by outputting an unusual token during sampling. For LLaMA models, with their unique tokenization of numbers, added spaces have the opposite effect. Each digit and space is already assigned its own token, and space tokens become nuisance inputs, adding to the sequence length without simplifying the sequence's structure and potentially making the sequence out-of-distribution to the model.

3.9.3 Rescaling

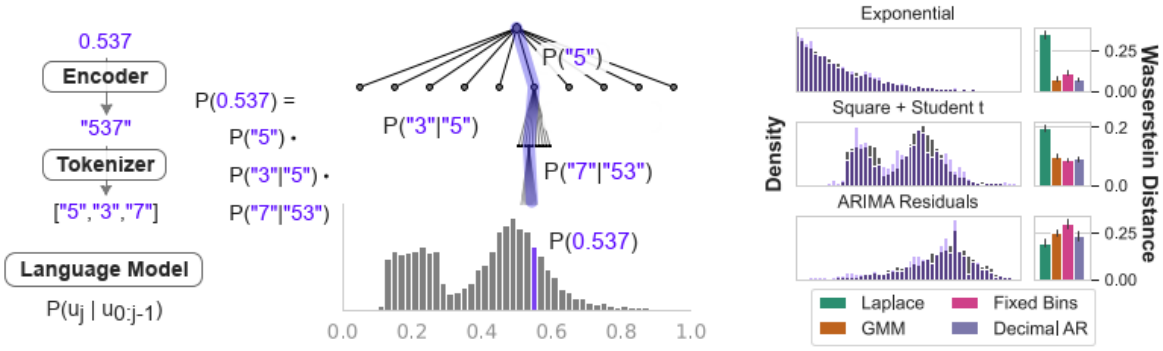
Rescaling To avoid wasting tokens when the inputs are very large, we scale values down so that the α-percentile of rescaled time series values is 1. We avoid scaling by the maximum value so that the LLM can see some fraction of examples $(1 - \alpha)$ where the number of digits changes and reproduce this behavior in its outputs to produce larger values than it has seen.

3.9.4 Sampling / Forecasting

Sampling / Forecasting To forecast, draw many samples (e.g. 20) from the LLM and use the statistics of the samples at each time step to construct a point estimate (e.g. as the median) or probabilistic forecast (e.g. as quantiles). To control sampling, we use temperature scaling, logit bias, and nucleus sampling.

3.9.5 Continuous likelihoods Modeling

Continuous likelihoods Modeling sequences of individual digits has additional benefits beyond good samples. With n digits of precision in base B^n, each sequence of n digits after the decimal place corresponds to one of Bn possible bins [Figure 12-3], each with width B^{-n}. As each distribution $p_\theta(u_j|u_{0:j-1}; 0)$ is a softmax over possible digits, we can view the distribution over each individual number as a hierarchical softmax , with $p(u_1, \ldots, u_n) = p(u_n|u_{n-1}, \ldots, u_0)p(u_0|u_1)p(u_0)$ Though a language model's probability distribution is discrete, we can easily adapt it to provide a continuous density by placing a uniform distribution in each bin.



[Figure 12-3] Left: Autoregressive models over sequences of digits act like hierarchical softmax distributions over the corresponding numbers. Right: Using simple autoregressive models (e.g. RNNs) trained on a string representation of numbers. [11]

Enumerating each of the countably infinite numbers that the model can produce (because the model can output an arbitrary number of digits before the decimal point) with an index $K \in \mathbb{N}$ each with probability p_k, we can write out the distribution as a mixture of disjoint uniform distributions over the bins $p(x) = \sum_{K\in\mathbb{N}} p_k U(x)$ where $U_k(x) = \mathrm{B}^n \mathbb{I}_{x\in[\mathrm{B}^{-n}k, \mathrm{B}^{-n}(k+1)]}$. Therefore, if a given data point lies in bin k, its continuous log likelihood is $\log \log p(x) = \log p_k + \mathrm{n}\log B$. Finally, to obtain the likelihood $\log p(z)$ in the original input space, we add a change of variables factor $\log \left|\frac{dx}{dz}\right|$ where $\mathrm{z} \rightarrow x = s(z)$ is the rescaling operation in the pre-processing. As a result, the exponentially large number of bins and exponentially small bin widths enabled by our construction make it surprisingly efficient to represent flexible and high-resolution continuous distributions with LLMs, despite using a discrete tokenization of numbers.

Summary

Across single and hybrid models, several approaches are popularly applied in the training process. Each model has its own advantages [Table 1] for achieving better prediction results. In addition, supporting information such as technical indicators, news data, or Fourier components is also very useful for prediction tasks. Moreover, data preprocessing is another effective method to identify trends or dependencies in the data and further enhance prediction performance. Overall, the combination of suitable models, enriched feature inputs, and effective preprocessing strategies can significantly improve the accuracy and reliability of time series prediction.

Reference#	model	Pros
[1]	CNN-BiLSTM-Attention CNN: extract spatial features BiLSTM: capture temporal dependencies Attention: weight value of the extracted features	Hybrid model BiLSTM effectively captures both past and future context in sequential data, leading to more accurate and comprehensive predictions than traditional LSTM models.
[2]	OPTIMAL LSTM Optimize architecture for traditional LSTM	Applying Technical Indicators into Deep Learning Model to enhance the accuracy of prediction.
[3]	HDFM CEEMDAN: empirical mode decomposition VMD: Variational Mode Decomposition GRU	Hybrid model Use CEEMDAN to decompose data And implement VMD to decompose the highest frequency data again then input to GRU then weigh the different frequency data to get the higher accuracy prediction.
[4]	LightGBM GRU+LSTM LR: Logistic Regression (LR) is used as a robust baseline model.	hybrid deep learning model that combines LightGBM, LSTM and GRU layers with a hyperparameter tuning mechanism
[5]	BiLSTM-MTRAN-TCN TCN: capture sequence dependencies and improve the model's generalization ability	Hybrid model Replaces the original decoder part with TCN layer and fully connected layer has high accuracy and generalization ability and does not have timeliness issues.
[6]	MVL-SVM support MVL-SVM support vector machine and multi-view learning machine and multi-view learning	Hybrid model based on multi-view heterogeneous data (News keywords) can predict stock price movement more accurately
[7]	improved Transformer and combining with feature fusion, self-attention, noise filtering	Hybrid model feature fusion on the input data and adaptive time window, weight adjustment mechanism
[8]	GANs: generate synthetic stock price data Attention mechanisms (VAE) selectively concentrate on the important features and patterns	Hybrid model: Adaptive impact of news per day Technical indicator and Fourier component into VAE then use important features to train GANs to generate accurate results.
[9]	Markov Decision Process CNN+RNN+Attention	Hybrid model: This paper focuses on High-frequency trading, adaptive CNN+RNN to extract features from stock price and attention to combine to then input to MDP to make active decisions to get better win rate.
[10]	Tokenizing + Self-attention	Prior to using attention model to train, the input data use Tokenizing technic to divide into sub-data tokens and project onto embedding vectors, thereby facilitating the model's comprehension of its data meaning to achieve more accurate data prediction.
[11]	LLM (GPT-3, GPT-4)	The model utilizes existing LLM model to predict data w/o training any extra model and can get the well prediction results.
[12]	N-BEATS Fully connect network	Just use fully connect network to stack up the empirical mode, the residual part will transfer to next block to training and can optimize the prediction result, and the result is interpretability.

[Table 1]

References:

[1] Zheng, Haotian, et al. “Predicting financial enterprise stocks and economic data trends using machine learning time series analysis.” *Applied and Computational Engineering* 87 (2024): 26-32. ewadirect.com

[2] Agrawal, Manish, Asif Ullah Khan, and Piyush Kumar Shukla. “Stock price prediction using technical indicators: a predictive model using optimal deep learning.” *Learning* 6.2 (2019): 7. researchgate.net

[3] Y. Li, L. Chen, C. Sun, G. Liu, C. Chen and Y. Zhang, "Accurate Stock Price Forecasting Based on Deep Learning and Hierarchical Frequency Decomposition," in IEEE Access, vol. 12, pp. 49878-49894, 2024

[4] M. Sui, C. Zhang, L. Zhou, S. Liao and C. Wei, "An Ensemble Approach to Stock Price Prediction Using Deep Learning and Time Series Models," 2024 IEEE 6th International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, 2024, pp. 793-797

[5] S. Wang, "A Stock Price Prediction Method Based on BiLSTM and Improved Transformer," in IEEE Access, vol. 11, pp. 104211-104223, 2023,

[6] Long, Wen, et al. "A hybrid model for stock price prediction based on multi-view heterogeneous data." *Financial Innovation* 10.1 (2024): 48.

[7] Yao, Y. (2025). Stock Price Prediction Using an Improved Transformer Model: Capturing Temporal Dependencies and Multi-Dimensional Features. *Journal of Computer Science and Software Applications*, *5*(2). https://doi.org/10.5281/zenodo.14832392

[8] Li, S., Xu, S. Enhancing stock price prediction using GANs and transformer-based attention mechanisms. *Empir Econ* **68**, 373–403 (2025).

[9] Cao , G. ., Zhang , Y. ., Lou , Q. ., & Wang , G. . (2024). Optimization of High-Frequency Trading Strategies Using Deep Reinforcement Learning. Journal of Artificial Intelligence General Science (JAIGS) ISSN:3006-4023, 6(1), 230–257. https://doi.org/10.60087/jaigs.v6i1.247

[10] Zhu, Zhuohang, et al. "Tokenizing Stock Prices for Enhanced Multi-Step Forecast and Prediction." *International Conference on Neural Information Processing*. Singapore: Springer Nature Singapore, 2024.

[11] Gruver, Nate, et al. "Large language models are zero-shot time series forecasters." Advances in Neural Information Processing Systems 36 (2023): 19622-19635.

[12] Oreshkin, Boris N., et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting." *arXiv preprint arXiv:1905.10437* (2019)

[13] Shubham P (2019) All you need to know about variational autoencoder. https://blog.bayeslabs.co/2019/06/04/All-you-need-to-know-about-Vae.html