A Survey of Time Series Data Forecasting Methods Based on Deep Learning

Abstract：Time Series Forecasting (TSF) involves predicting future values and trends of data at specific points or periods by analyzing historical patterns, such as trends and seasonality. Generated by sensors, time series data play a vital role in various domains, including finance, healthcare, energy, transportation, and meteorology. With the advent of IoT sensors, traditional machine learning approaches struggle to handle massive time series datasets. Recently, deep learning algorithms, exemplified by convolutional neural networks (CNNs), recurrent neural networks (RNNs), and Transformer models, have made significant progress in time series forecasting tasks. This paper reviews the common features of time series data, relevant datasets, and evaluation metrics for models. It also conducts experimental comparisons of various forecasting algorithms, focusing on time and algorithmic architectures. Major deep learning-based time series forecasting methods are introduced and compared. Finally, challenges and future research directions in applying deep learning to time series forecasting are discussed.

Keywords：Deep Learning; Time Series Forecasting; Recurrent Neural Networks; Gated Recurrent Units; Transformer Model

1 Research Background

Time series data exist widely in domains like finance, healthcare, energy, transportation, and meteorology and are easily accessible. However, with the widespread use of sensing devices and advancements in data processing, time series data are being generated at an explosive rate. Analyzing these data is crucial for extracting valuable information, such as weather predictions, traffic flow forecasts, financial analysis, flu trend monitoring, medical responses, and system workload management[1].

Challenges in time series forecasting include sequence length and parallelization issues. Traditional statistical models, such as Support Vector Machines (SSM) and Autoregressive Models (AR), require manual configuration of seasonal and trend components[2-3], limiting their efficiency and accuracy with large-scale datasets. Deep neural networks (DNNs) offer an alternative due to their ability to extract high-level features and identify complex patterns within and across time series with minimal manual effort. However, DNNs require extensive training data due to their reliance on fewer structural assumptions.

Convolutional Neural Networks (CNNs), as one of the most representative network architectures in deep learning[4], hold broad application prospects in this field. Compared to traditional methods, CNNs demonstrate superior capabilities in feature extraction and information mining. While CNNs are highly effective for processing image data, they face limitations when applied to time series data, such as sequences, speech, and text. Specifically, CNNs struggle to capture long-range dependencies in time series. Although increasing the depth of convolutional layers can expand the receptive field, it often remains insufficient for modeling long-term dependencies within sequences.In 1990, Jeffrey Elman introduced the foundational concept and structure of Recurrent Neural Networks (RNNs) in his paper Finding Structure in Time. Elman's model introduced the concept of a hidden layer (or internal state), which stores information about previous elements in a sequence and uses it to predict the next element. This breakthrough opened the door for RNN applications in natural language processing, time series analysis, and other domains. Subsequently, RNNs evolved to address issues like vanishing and exploding gradients in long-sequence processing through variants such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU).In 2017, Vaswani et al. proposed the Transformer architecture, introducing the self-attention mechanism—a key innovation that enables exceptional performance in processing sequence data. The Transformer's self-attention mechanism captures long-term dependencies, addressing limitations of traditional RNNs in long-range prediction and parallel computation. As a result, Transformers have achieved remarkable performance in time series forecasting tasks.

In modern domains, time series forecasting methods are already well-established. For instance, Facebook's visual interactive network (VIN) employs bidirectional LSTM to effectively capture temporal information in videos[5]. Chinese researchers have also achieved significant

results in tasks such as image captioning and video classification. For example, the Chinese Academy of Sciences proposed the spatio-temporal attention network (STAN), which uses bidirectional GRU to enhance video classification accuracy[6].Photovoltaics (PV), one of the most promising renewable energy sources, requires accurate power forecasting to ensure the safe operation and economic integration of PV systems in smart grids. Mohamed Abdel-Nasser et al. proposed an LSTM-RNN-based method for predicting PV system output power, providing a useful tool for smart grid planning and control[8].Qiang Cui et al. introduced a Multi-View Recurrent Neural Network (MV-RNN) model to handle sequence recommendations and cold-start issues. By integrating visual and textual information, MV-RNN mitigates cold-start problems, dynamically captures user interests, generates personalized ranking lists, addresses missing modality issues, and alleviates cold-start challenges[8].Shaowei Pan et al. proposed a hybrid model (CNN-LSTM-SA), combining Convolutional Neural Networks (CNNs), LSTM networks, and self-attention mechanisms (SA). This model achieved optimal performance in capturing fundamental trends and predicting specific values for oil well production[9].In the field of speech recognition, RNNs and their variants have achieved significant breakthroughs. For example, Microsoft's deep neural network (DNN) combined with LSTM demonstrated excellent performance in speech recognition competitions. Domestic companies such as iFLYTEK and Sogou have also adopted RNN-based technologies to improve speech recognition accuracy[10].In hydrological time series forecasting, Muhammad Waqas et al. demonstrated the effectiveness of RNNs and LSTMs in modeling nonlinear and time-varying hydrological systems, making them a research hotspot[11].The Transformer's self-attention mechanism excels in capturing long-range dependencies, offering superior temporal modeling capabilities and advantages in handling time series data[12]. Liu et al. proposed a de-stationary framework to address over-stabilization issues in processing raw time series data[13]. Additionally, Liu et al. recently introduced an inverted input approach that reassigns roles between the attention module and the feedforward neural network[14], effectively enhancing SOTA model performance.In 2023, Zhang et al. proposed a multi-scale pyramid Transformer model called MTPNet[15]. The use of multi-layer Transformer structures with different scales has solved the problem of time dependent modeling for fixed or constrained scales.

2 Evaluation Metrics and Datasets for Time Series Forecasting

2.1 Evaluation Metrics for Time Series Forecasting

Evaluation metrics are tools used to assess and analyze the performance of time series forecasting models, serving as key criteria for measuring model performance. Common evaluation metrics for time series forecasting include:

Mean Squared Error (MSE): This measures the average squared difference between predicted and actual values, reflecting the overall error between the predictions and actual outcomes. It is calculated as follows:

$$MSE\ (y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

Root Mean Squared Error (RMSE): This is the square root of MSE, assigning higher weights to larger errors and emphasizing the stability of the prediction results. It is calculated as follows:

$$RMSE\ (y,\hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

Mean Absolute Error (MAE): This represents the mean absolute difference between predicted and actual values, reducing the influence of outliers. It is calculated as follows:

$$MAE\ (y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(|\ y_i - \hat{y}_i\ |) \tag{3}$$

Mean Absolute Percentage Error (MAPE): This metric considers the relative magnitude of the actual values, avoiding the cancellation effect of positive and negative errors. It is calculated as follows:

$$MAPE\ (y,\hat{y}) = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4}$$

Symmetric Mean Absolute Percentage Error (SMAPE): This is a modification of MAPE that avoids excessively large values when the actual values are very small. It is calculated as follows:

$$SMAPE\ (y,\hat{y}) = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\ \hat{y}_i\ |)/2} \tag{5}$$

Coefficient of Determination (R^2): Also known as the goodness of fit, this metric divides the explained variance by the total variance to measure the proportion of variance in the dependent variable explained by the independent variables. It is calculated as follows:

$$R^2(y,\hat{y}) = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y_i})^2} = 1 - \frac{MSE}{Var} \tag{6}$$

In the formulas (1) to (6) mentioned above, y is the true value, $\hat{y}$ is the predicted value, $\bar{y}$ is the mean of y, and Var is the variance .

Except for R^2, all the evaluation metrics mentioned above are better when their values are smaller. The choice of evaluation metric depends on the specific situation. Typically, a combination of multiple metrics is used to comprehensively analyze the model, enabling deeper insights. Due to the differences in the characteristics and application focus of Recurrent Neural Network (RNN) models and Transformer models when handling time series data, researchers prioritize different metrics when evaluating algorithm performance.

RNN models, with their memory capability, excel at capturing long-term dependencies in time series. Consequently, evaluation metrics for RNNs often emphasize the ability to model the overall structure of the time series and adaptability to various tasks. Researchers may use a diverse range of evaluation metrics to thoroughly assess the performance of these algorithms.On the other hand, Transformer models, with their self-attention mechanism allowing for parallel computation, are more efficient and focus heavily on predictive accuracy. As such, researchers commonly use MAE (Mean Absolute Error) and MSE (Mean Squared Error) to measure the performance of Transformers.

2.2 Datasets

2.2.1 ETT

The ETT dataset, provided by the State Grid Corporation of China, consists of minute-level recordings of transformer oil temperatures from two counties in the same province during 2016–2018. Each dataset contains 1,051,200 data points. To explore long-term prediction granularity, the dataset was divided into subsets based on sampling intervals of 15 minutes and 1 hour, resulting in four subsets: ETTm1, ETTm2, ETTh1, and ETTh2. These subsets contain 69,680 and 17,420 data points, respectively. Each data point includes seven features, comprising the target variable (oil temperature) and six types of power load features.

2.2.2 ECL

ECL(Electricity Consuming Load):Electricity Consumption Load dataset from 2012–2014.

2.2.3 Traffic

Traffic:Traffic dataset from California's Department of Transportation (2015–2016).

2.2.4 Weather

Weather:The weather dataset, provided by the meteorological station of the Max Planck Institute for Biogeochemistry, records 21 meteorological indicators such as air pressure, temperature, and humidity collected every 10 minutes from 2020 to 2021.

2.2.5 ILI

ILI: The influenza dataset, provided by the Centers for Disease Control and Prevention in the United States, records the ratio of influenza like disease patients to the total number of patients per week from 2002 to 2021.

2.2.6 TE

Tennessee Eastman (TE) is a representative chemical process proposed by an American chemical company, consisting of a gas-liquid separator, a circulating compressor, a stripper, a condenser, a reactor, and other components. The TE chemical process can simulate 21 types of faults in industrial production processes. These faults are mainly divided into 6 types, including constant position, sticking, step, random variable, slow drift, and five unknown faults. The variable parameters of this process include 41 measured variables (XMEAS (1) - XMEAS (41)) and 12 manipulated variables (XMV (1) - XMV (12)), for a total of 53 observed variables.

3 Time series prediction model based on deep learning

3.1 RNN

The RNN model consists of the following three main components:

(1) Input Layer: Receives input data and passes it to the hidden layer. The input includes not only static data but also historical information from the sequence.

(2) Hidden Layer: The core component that captures temporal dependencies. The output of the hidden layer depends on both the current input and the hidden state from the previous timestep.

(3) Output Layer: Generates the final prediction based on the output of the hidden layer.

The structure of the RNN is illustrated in the diagram below:

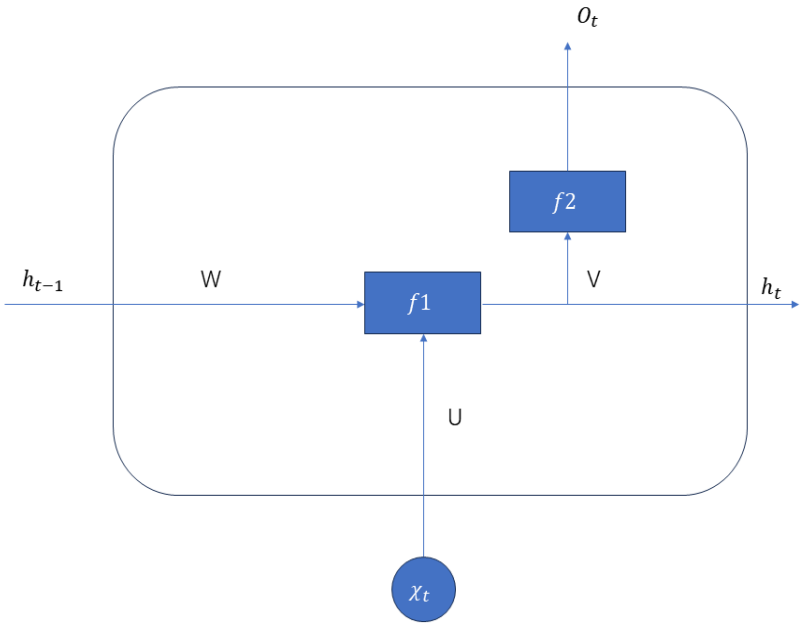


Fig.1 RNN structure diagram

The diagram shows the basic structure of an RNN, where o_t represents the output information, h_t represents the hidden layer output at the current timestep, h_{t-1} represents the hidden layer output from the previous timestep, and x_t represents the current input. Functions f_1 and f_2 are activation functions, while W, U, V represent weight matrices. RNNs work by continuously cycling the same neuron over time. The calculation for the current timestep is given by:

$$h_t = f_1(Ux_t + Wh_{t-1} + b) \tag{7}$$

$$o_t = f_2(Vh_t + c) \tag{8}$$

The working principle of RNNs can be summarized in the following steps: At each timestep, the RNN unit receives the input x_t for the current timestep and the hidden state h_{t-1} from the previous timestep. Based on these, the hidden layer computes a new state h_t using a nonlinear function (such as Tanh or ReLU). The output layer then generates the final output o_t using another weight matrix and activation function.

Although RNNs are effective for processing sequential data, they suffer from issues like vanishing and exploding gradients. For long sequences, the gradients may become very small (vanish) or excessively large (explode) due to repeated multiplication. To address these problems,

variants of RNNs, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU), have been developed.

3.2 Long Short-Term Memory (LSTM)

LSTM is a specialized RNN architecture proposed by Hochreiter and Schmidhuber in 1997. It was designed to overcome the gradient vanishing and exploding problems encountered in standard RNNs when handling long sequences. The core of an LSTM is its sophisticated gating mechanism, which controls the flow of information in and out of the unit. A typical LSTM unit comprises the following components: forget gate, input gate, cell state, and output gate. Below is its structure diagram:

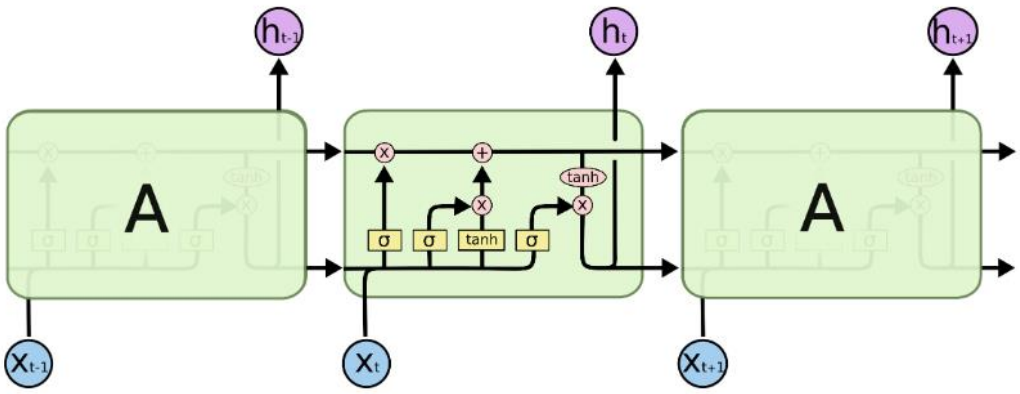


Fig. 2 LSTM Structure Diagram

3.2.1 **Forget Gate**

In an LSTM, the first step is determining which information should be filtered out from the cell state by the forget gate. This operation is achieved through the forget gate's structure. The forget gate reads the previous output h_{t-1} and the current input x_t, applies a Sigmoid nonlinear transformation, and outputs a vector f_t. Each value in this vector ranges from 0 to 1, where 1 indicates complete retention and 0 indicates complete discard. This vector is then multiplied element-wise with the cell state C_{t-1}.For example, in a language model, the cell state may encode the gender information of the subject in the current sentence, ensuring the correct pronoun is selected. When a new subject is identified, the forget gate removes the prior subject's gender

information to make room for the new information.

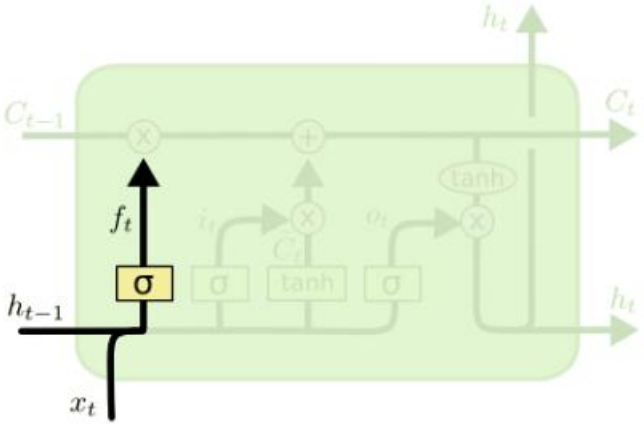


Fig. 3 Forgetting Gate Structure Diagram

The mathematical formula for the forget gate is as follows:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{9}$$

here (W_f) and (b_f) are the weight matrix and bias vector, (h_{t-1}) is the previous hidden state, (x_t) is the current input, and σ is the Sigmoid activation function.

3.2.2 Input Gate

The information update mechanism in Long Short-Term Memory (LSTM) networks involves the following two steps to determine how new information is stored in the cell state:

(1) Input Gate activation function. This is a layer composed of a sigmoid activation function that determines which values will be updated to the cellular state. This layer outputs a vector between 0 and 1, where each element corresponds to the updated weight of the corresponding element in the candidate cell state.

(2) Generation of candidate cell states. This layer, formed by a tanh activation function, creates a vector of potential new information containing values that might be integrated into the current cell state. This newly generated vector is then multiplied element-wise by the output of the input gate to determine the actual values added to the cell state.

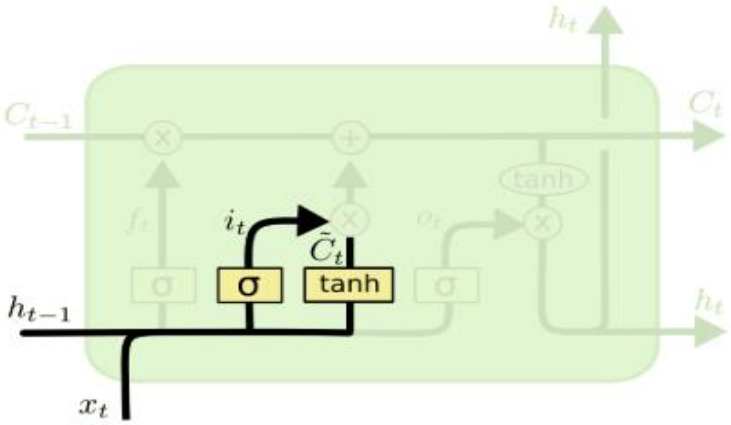


Fig. 4 Input door structure diagram

The mathematical expressions for these two steps are as follows:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{10}$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{11}$$

here (W_i)and (b_i)， (W_c)and（ b_c） are the parameter matrices and bias vectors, (h_{t-1}) is the hidden state from the previous timestep, (x_t) is the current input, σ and tanh are activation functions.

The next step updates the cell state C_t by combining the information to be forgotten and the newly added information:

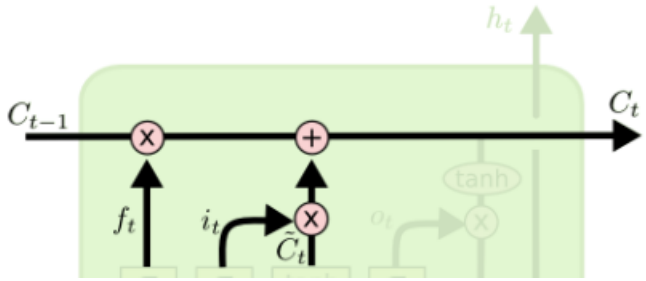


Fig.5 Update cell structure diagram

LSTM cells need to multiply the previous state C_{t-1} with f_t, discard the information that needs to be discarded, and then add $i_t * \tilde{C}_t$. This is the new output state C_t.

3.2.3 **Output Gate**

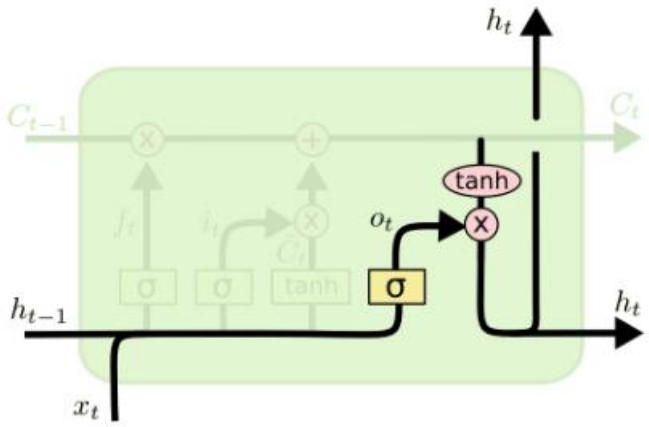


Fig.6 Output Gate Structure Diagram

The mathematical formula for the output gate is as follows:

$$o_t = \sigma(W_0[h_{t-1}, x_t] + b_0) \tag{12}$$

$$h_t = o_t * \tanh(C_t) \tag{13}$$

here (W_0)and (b_0) are the parameter matrices and bias vectors, (h_{t-1}) is the previous timestep's hidden state, (x_t) is the current input, and σ is the sigmoid activation function.

3.3 Gated Recurrent Units (GRU)

GRU is a simplified variant of LSTM that retains the gating mechanisms (update and reset gates) to control the flow of information while omitting the separate memory cell. GRUs have fewer parameters than LSTMs, resulting in higher computational efficiency and, in some cases, similar or better performance. The core components of GRU include:

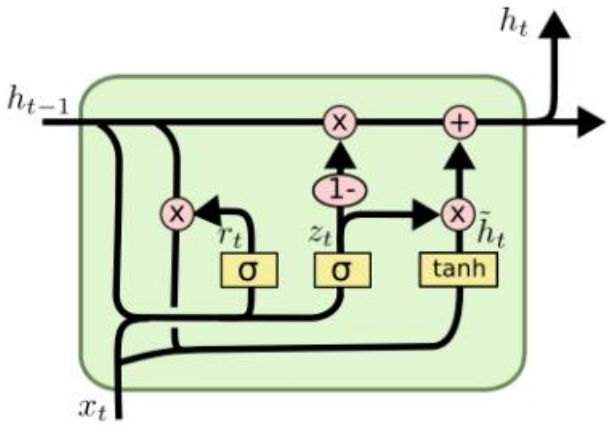


Fig.7 GRU structure diagram

3.3.1 Update Gate

The update gate determines how much of the previous timestep's hidden state should be retained in the current hidden state. It outputs values between 0 and 1, where higher values indicate greater retention of past information and lower values suggest reliance on current input. The formula is:

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{14}$$

here (W_z) and (b_z) are the parameter matrices and bias vectors,(h_{t-1}) is the previous hidden state, (x_t) is the current input, and σ is the sigmoid activation function.

3.3.2 Reset Gate

The reset gate determines the extent to which the previous hidden state influences the computation of the candidate hidden state. When the reset gate output is close to 0, most of the previous information is ignored; when it is close to 1, more past information is retained. The formula is:

$$r_t = \sigma(W_t[h_{t-1}, x_t] + b_t) \tag{15}$$

here（W_r）and（b_r） are the parameter matrices and bias vectors.

At each time step, the GRU unit processes information through the following steps:

(1) Calculate update and reset gates

According to the above formula, calculate the update gates (z_t) and (r_t) respectively, and the outputs of these gates will control the update of hidden states and the degree of preservation of historical information.

(2) Calculate candidate hidden states

GRU uses reset gates to control the degree of dependence on previously hidden states and calculate candidate hidden states.

The calculation formula for candidate hidden states is:

$$\tilde{h}_t = \tanh(W[r_t h_{t-1}, x_t] + b) \tag{16}$$

Among them, ($r_t * h_{t-1}$) represents combining the hidden state of the previous time step (h_{t-1}) with the reset gate (r_t) to control its degree of influence, and (W) and (b) are the parameter

matrix and bias.

(3) Update Hidden State

The final step is to use the update gate (z_t) to calculate the current hidden state (h_t):

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{17}$$

Here, $(1 - z_t) * h_{t-1}$ controls the retention of past information, and $z_t * \tilde{h}_t$ introduces new information.This formula ensures that the current hidden state incorporates both historical and new information effectively.

3.4 Bi-LSTM Model

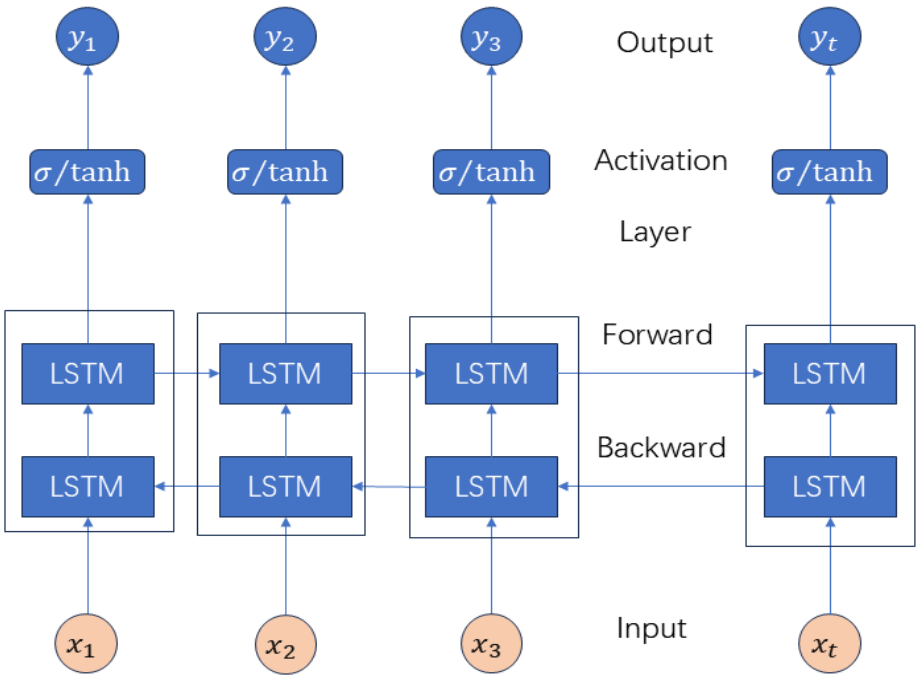


Fig. 8 Bi LSTM Structure Diagram

The Bi-LSTM (Bidirectional Long Short-Term Memory) network is an improved version of the LSTM network. It combines two LSTMs: one processes the sequence from the beginning to the end, while the other processes it in reverse, from the end to the beginning. This architecture excels in several tasks compared to standard LSTM networks.Bi-LSTM is a time-recurrent neural

network that stacks forward and backward LSTM layers together. The output is determined by the hidden states of these two LSTM layers. Bi-LSTM combines the traditional forward and backward time sequences, leveraging LSTM's sensitivity to sequence order to construct a bidirectional network.The concatenated vector of outputs from the forward and backward processes provides the complete hidden representation of Bi-LSTM, as shown below:

$$h_t = h_t^f \oplus h_t^b \tag{18}$$

Here, h_t^f is the output from the forward LSTM layer, and h_t^b is the output from the backward LSTM layer. These outputs are combined using element-wise summation.The internal structure of Bi-LSTM cells is identical to that of standard LSTMs and is thus not elaborated further.

3.5 Transformer Model

The Transformer is a deep learning architecture primarily used for natural language processing (NLP) and other sequence-to-sequence tasks. It was first proposed by Vaswani et al. in 2017[16]. The key innovation of the Transformer architecture is the self-attention mechanism, which allows it to excel in processing sequential data.The Transformer employs a self-attention-based encoder-decoder structure.

(1) **Encoder**: Composed of stacked identical layers, each layer includes two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feedforward network. Normalization layers and residual connections are applied to the input and output of the multi-head self-attention module.

(2) **Decoder**: The decoder generates the output sequence using the representation produced by the encoder. Similar to the encoder, the decoder is composed of stacked identical layers. Each decoder layer adds a third sub-layer, which performs multi-head attention over the encoder's output. Residual connections and normalization layers are applied to each sub-layer.The following is the architecture diagram of Transformer.

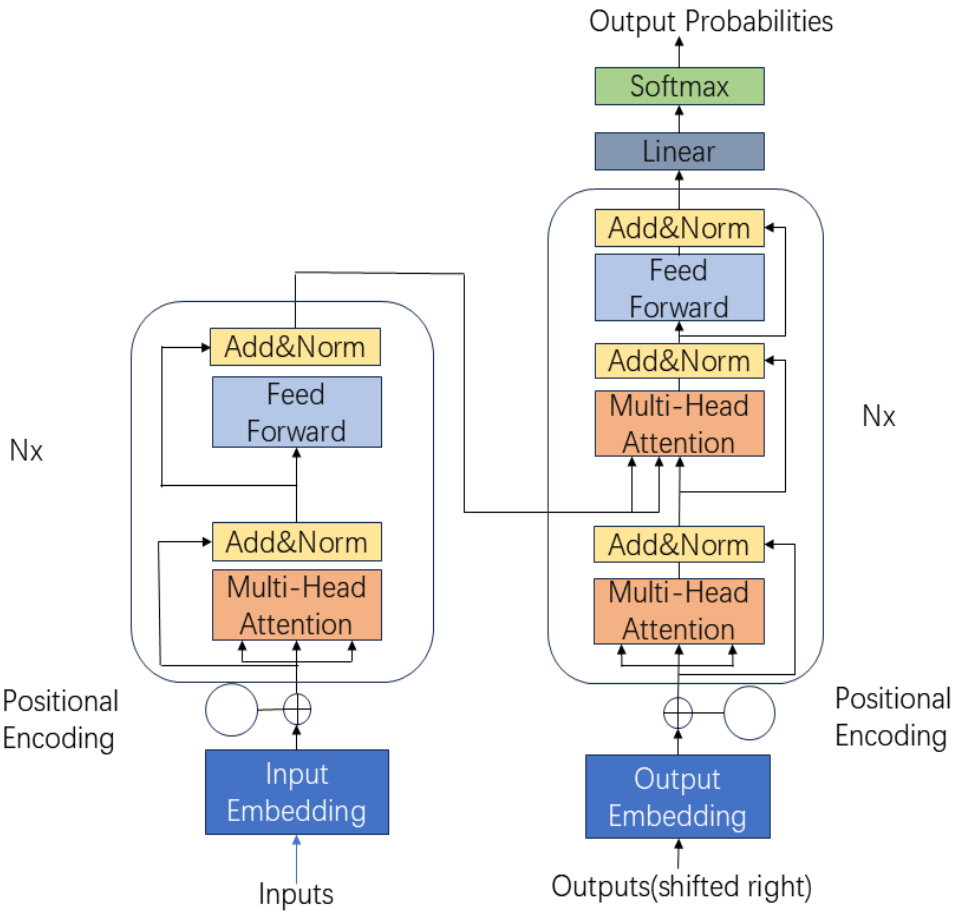


Fig.9 Transformer architecture diagram

3.5.1 Self-Attention Mechanism

The self-attention mechanism is the core concept of the Transformer. It enables the model to consider all positions in the input sequence simultaneously, unlike recurrent or convolutional neural networks that process sequentially. The self-attention mechanism assigns varying attention weights to different parts of the input sequence, thereby capturing semantic relationships more effectively.The mathematical expression for self-attention is as follows:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{19}$$

Firstly, extract the query vector (Q), key vector (K), and value vector (V) from the embedded vector. Next, determine a score for each vector: the score is equal to $Q \bullet K$. Score normalization (division $\sqrt{d_k}$) is used for gradient stability. Next, use the softmax activation function to process the scores. The weighted score of each input vector is obtained by taking the

softmax dot product value. Sum up to produce the final result.

3.5.2 Multi-Head Attention Mechanism

The Transformer model employs a multi-head self-attention mechanism to enhance its ability to capture dependencies among elements in a sequence. The core principle of the attention mechanism is that each token in a sequence can aggregate information from other tokens,enabling the model to better understand contextual relationships.This is achieved by mapping a query, a set of key-value pairs, and an output (each represented as vectors) into an attention function. The output is computed as a weighted sum of the values, where the weights are determined by a compatibility function between the query and the corresponding keys.Multi-head attention is equivalent to combining multiple scaled dot-product attention mechanisms. It effectively parallelizes the processing of the query (Q), key (K), and value (V) vectors, resulting in a final output that integrates information from different attention heads.

3.5.3 Positional Encoding

Since the Transformer model does not rely on recursion or convolution, it requires a method to capture the relative or absolute position of tokens within a sequence to effectively utilize sequential order. Positional encoding is introduced at the input level of the encoder and decoder stacks.These positional encodings are added to the input embeddings, sharing the same dimensional space. They are calculated using sine and cosine functions at different frequencies as follows:

$$PE(t)_i = \begin{cases} sin\left\{\dfrac{t}{10000^{2k/d}}\right\}, i = 2k \\ cos\left\{\dfrac{t}{10000^{2k/d}}\right\}, i = 2k+1 \end{cases} \tag{20}$$

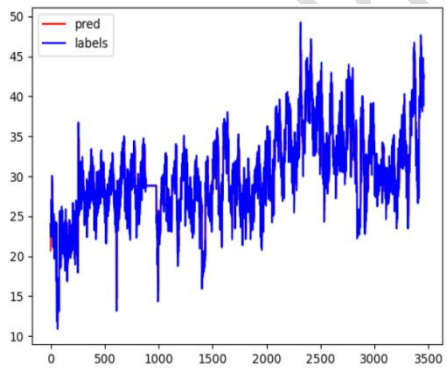
Here, t is the position in the sequence, d is the vector dimension, and k is the natural number used for indexing. By mapping each position to a unique frequency using sine and cosine functions, and converting the frequency into an element in the embedding vector using the corresponding sine and cosine functions, the model can capture the position information when processing the input sequence.

3.6 Comparison of Deep Learning Models

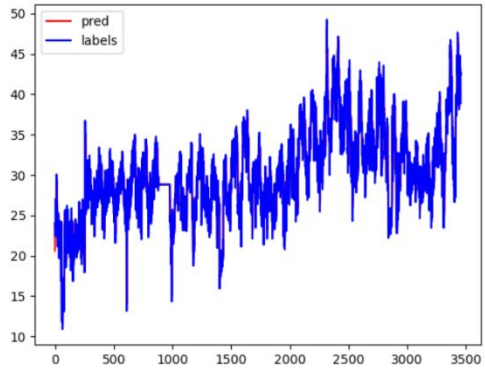
Datasets: This study conducted experiments using the ETTm2, Electricity, Traffic, and Weather datasets. Other datasets were excluded due to insufficient periodicity, seasonality, or data volume.

Experiment Details: To ensure consistency, all models used an input sequence length of 24 and a prediction length of 1. For the ETTm2, Electricity, and Weather datasets, the first variable was selected for prediction, representing the high-useful load of an electric transformer oil temperature, a user's hourly electricity consumption, and atmospheric pressure, respectively. For the Traffic dataset, the third variable was selected, representing the hourly road occupancy rate recorded by a sensor. All models were trained using the Adam optimizer, with MSE (Mean Squared Error) and MAE (Mean Absolute Error) as evaluation metrics. PyTorch was used for implementation.

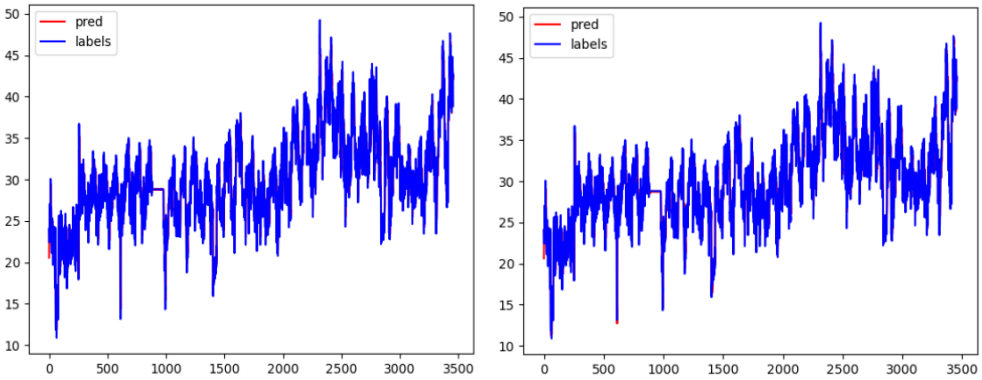
Fig.10 shows the prediction curves of four models on the ETTm2 dataset, where the horizontal axis represents the dataset, the vertical axis represents the numerical size, the blue curve represents the actual data value, and the red curve represents the predicted value.



(a) RNN model



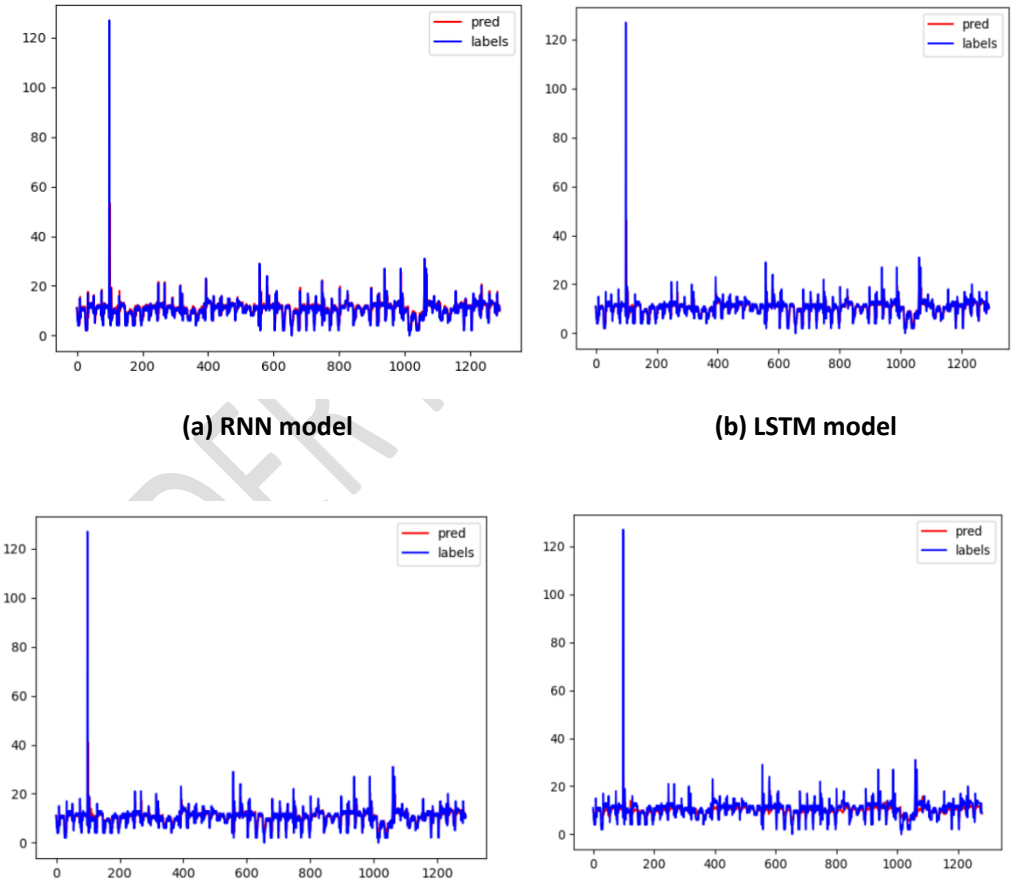
(b) LSTM model



(c) GRU model **(d) Transformer model**

Fig.10

Fig.11 shows the prediction curves of four models on the ECL dataset, where the horizontal axis represents the dataset, the vertical axis represents the numerical size, the blue curve represents the actual data value, and the red curve represents the predicted value.



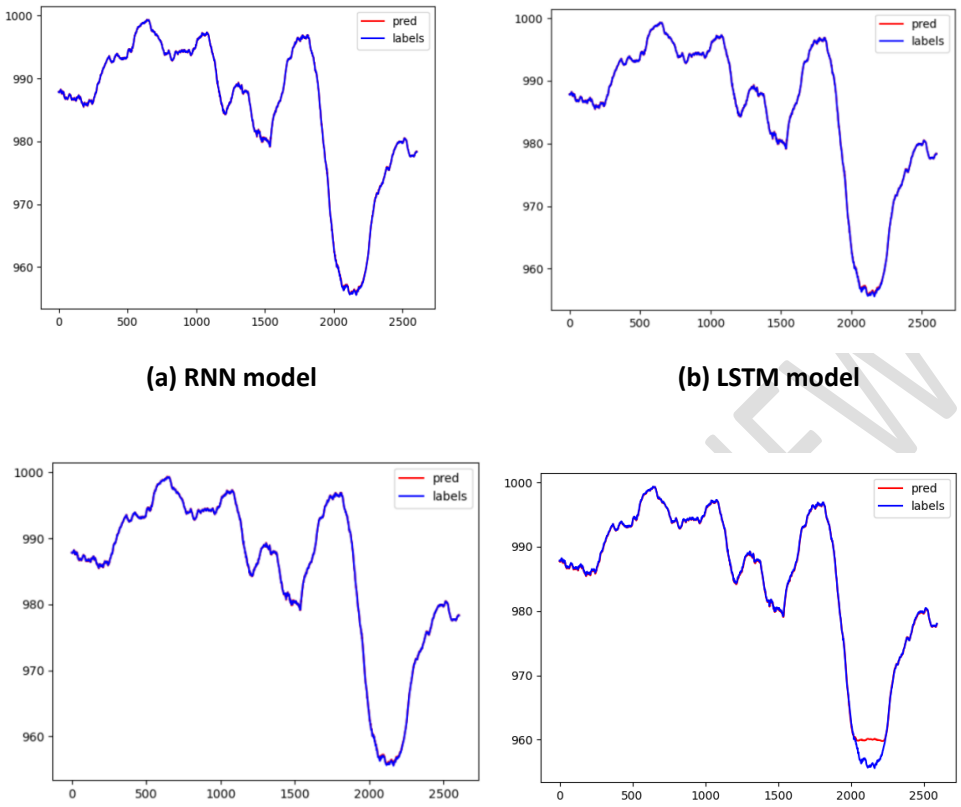
(a) RNN model **(b) LSTM model**

(c) GRU model **(d) Transformer model**

Fig.11

Fig.12 shows the prediction curves of four models on the Weather dataset, where the horizontal axis represents the number of test sets, the vertical axis represents the numerical size,

the blue curve represents the actual data value, and the red curve represents the predicted value.

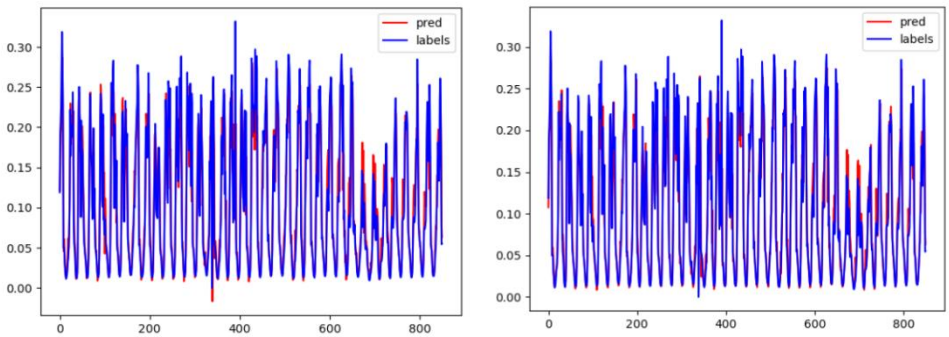


(a) RNN model **(b) LSTM model**

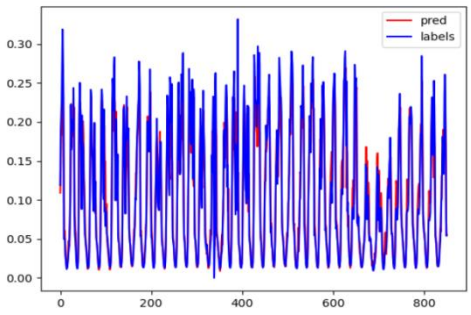
(c) GRU model **(d) Transformer model**

Fig.12

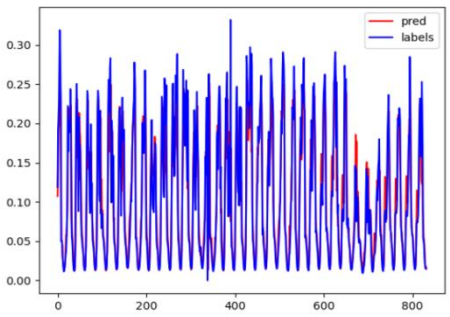
Fig.13 shows the prediction curves of four models on the Traffic dataset, where the horizontal axis represents the number of test sets, the vertical axis represents the numerical size, the blue curve represents the actual data value, and the red curve represents the predicted value.



(a) RNN model **(b) LSTM model**



(c) GRU model



(d) Transformer model

Fig.13

Compare and analyze the performance of the four deep learning models mentioned above (as shown in Table 1).

Table 1 Comparison of Univariate Prediction Performance of Four Deep Learning Models

Model	Evaluating indicator	ETTm2	Electricity	Weather	Traffic
RNN	MSE	3.459	21.603	**0.007**	0.00112
	MAE	1.404	2.014	**0.060**	0.021
LSTM	MSE	3.480	19.821	0.008	0.00120
	MAE	1.414	**1.848**	0.066	**0.020**
GRU	MSE	3.454	**19.524**	0.007	**0.00110**
	MAE	1.402	1.889	0.062	0.021
Transformer	MSE	**3.418**	19.541	0.827	0.00122
	MAE	**1.399**	2.025	0.341	0.021

The experimental data of various models listed in Table 1 were analyzed and summarized in depth, and the conclusion is as follows: the Transformer model achieved the best performance on the ETTm2 dataset, with minimum MAE and MSE values. The LSTM model achieved the minimum MAE on the Electricity and Traffic datasets, i.e. the minimum tie error. The GRU model achieved the minimum mean square error (MSE) on the Electricity and Traffic datasets. The RNN model achieved the best performance on the Weather dataset, with minimum MAE and MSE values.

4 Summary and Outlook

4.1 Summary

Although traditional statistical modeling techniques incorporate structural assumptions into models, making them easier to understand, they often require independent modeling of time series data in modern predictive applications. This approach significantly increases labor and computational costs. Therefore, it is necessary to find more efficient techniques capable of simultaneously handling varying degrees of relationships among two or more variables.Deep learning techniques can accurately identify complex patterns within and across time series with relatively lower human resource requirements. However, these models rely on fewer structural assumptions, making them more challenging to interpret and often requiring larger training datasets to learn accurate models. Additionally, since different sample types exhibit distinct distribution patterns, a single fixed model cannot be universally applied, necessitating the use of multiple regression algorithms.This has led to innovative forecasting methods that combine traditional statistical models with deep learning. These hybrid approaches have significantly addressed the limitations of both techniques. In recent years, many deep neural network models for time series analysis have been proposed. These methods not only enable models to automatically extract features and learn complex temporal patterns but also apply assumptions like temporal smoothing, enhancing model interpretability.As research on neural network technologies continues to advance, deep learning has become one of the hottest research topics in machine vision. Based on a review of literature on time series forecasting and deep learning, this paper primarily explores four deep learning models for time series forecasting.

4.2 Outlook

Deep learning has achieved significant results in the field of time series prediction, and the future prospects are even more exciting. With the continuous advancement of technology, we have reason to believe that deep learning will play a more important role in temporal prediction, bringing revolutionary changes to various industries.

Firstly, at the algorithmic level, deep learning will be further optimized and improved. At present, models such as RNN, LSTM, and GRU have achieved good results in time series prediction. In the future, new deep learning models such as Transformers and graph neural networks are expected to better handle time-series data, improve prediction accuracy and stability.

Secondly, at the application level, deep learning will play an important role in more fields. For example, in the financial field, deep learning can be used for stock price prediction, risk management, etc; In the field of energy, it can be used for power load forecasting, new energy generation forecasting, etc; In the field of transportation, it can be used for traffic flow prediction, flight delay prediction, etc. With the continuous accumulation of data, the application of deep learning in time series prediction will become more widespread.

In addition, cross domain fusion of deep learning in temporal prediction will also become a development trend. For example, combining deep learning with statistics, chaos theory, etc. is expected to further improve the generalization ability and robustness of prediction models. At the same time, by combining domain knowledge, deep learning can achieve finer grained temporal prediction, providing decision-makers with more targeted recommendations.

Finally, with the continuous improvement of computing power, the real-time performance of deep learning in temporal prediction will be guaranteed. In the future, real-time prediction will become possible, providing more accurate and real-time decision support for various industries.

5 Conclusion

This paper presents a systematic review of time series forecasting methods based on deep learning. It first introduces the background, significance, and various methods of time series forecasting. Then, it provides a detailed overview of representative deep learning models in this domain, including RNN, LSTM, GRU, and Transformer models. Subsequently, it conducts comparative prediction experiments on public datasets and evaluates the performance of these models. Finally, it explores future research directions for deep learning in time series forecasting, offering valuable insights for further advancements in this field.

References

[1] ESLIN G P, AGON C. Time-series data mining[J]. ACM Computing Surveys (CSUR), 2012, 45(1): 1-34.

[2] GERS F A, SCHMIDHUBER J, CUMMIN S F. Learning to forget: Continual prediction with LSTM[J]. Neural computation, 2000, 12(10): 245 1-2471.

[3] DURBIN J, KOOPMAN S J. Time series analysis by state space methods[M]. 2nd edn. OUP Oxford, 2012.

[4] Shaowei P ,Bo Y ,Shukai W , et al.Oil well production prediction based on CNN-LSTM model with self-attention mechanism[J].Energy,2023,284

[5] Michael E N ,Bansal C R ,Ismail A A A , et al.A cohesive structure of Bi-directional long-short-term memory (BiLSTM) -GRU for predicting hourly solar radiation[J].Renewable Energy,2024,222119943-.

[6] Hanen B ,Ali A B ,Riadh I F .A Bi-GRU-based encoder–decoder framework for multivariate time series forecasting[J].Soft Computing,2024,28(9-10):6775-6786.

[7] Abdel-Nasser M ,Mahmoud K .Accurate photovoltaic power forecasting models using deep LSTM-RNN[J].Neural Computing and Applications,2019,31(7):2727-2740.

[8] Qiang C ,Shu W ,Qiang L , et al.MV-RNN: A Multi-View Recurrent Neural Network for Sequential Recommendation[J].IEEE Transactions on Knowledge and Data Engineering,2020,32(2):317-331.

[9] Shaowei P ,Bo Y ,Shukai W , et al.Oil well production prediction based on CNN-LSTM model with self-attention mechanism[J].Energy,2023,284

[10] Fang W ,12 ,Chen Y , et al.Survey on Research of RNN-Based Spatio-Temporal Sequence PredictionAlgorithms[J].Journal on Big Data,2021,3(3):97-110.

[11] Waqas M ,Humphries W U .A critical review of RNN and LSTM variants in hydrological time series predictions[J].MethodsX,2024,13102946-102946.

[12] WEN Q S, ZHOU T, ZHANG C L, et al. Transformers in Time Series: A Survey[C]. IJCAI 2023: 6778-6786.

[13] LIU Y, WU H X, WANG J M, et al. Non-stationary trans-formers: Exploring the stationarity in time series forecast-ing[J]. Advances in Neural Information Processing Sys-tems, 2022, 35: 988

1-9893.

[14] LIU Y, HU T, ZHANG H, et al. ITransformer: inverted transformers are effective for time series forecasting[C]. ICLR 2024.

[15] ZHANG Y F, WU R, DASCALU S M, et al. Multi-scale transformer pyramid networks for multivariate time series forecasting[J]. IEEE Access, 2024:1473 1-14741.

[16] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information pro-cessing systems, 2017, 30:5998-6008.