Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

LDformer: a parallel neural network model for long-term power forecasting*

Ran TIAN‡, Xinmei LI, Zhongyu MA, Yanxing LIU, Jingxia WANG Chu WANG

Department of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China

E-mail: tianran@nwnu.edu.cn; 2020211978@nwnu.edu.cn; mazybg@nwnu.edu.cn; lyanxing@nwnu.edu.cn; 2020222004@nwnu.edu.cn; 2020221992@nwnu.edu.cn

Received Nov. 3, 2022; Revision accepted Feb. 13, 2023; Crosschecked

Abstract: Accurate long-time series power load forecasting is very important in the decision-making operation of the power grid and power consumption management of customers and can ensure the power system's reliable power supply and the grid economy's reliable operation. However, most time series forecasting models do not perform well in long-term time series forecasting tasks with large amounts of data and high forecasting accuracy. To address this challenge, we propose a parallel time series forecasting model called LDformer. First, we combine the Informer with LSTM to obtain deep expression capability in the time series. Then, we propose a parallel encoder module to improve the robustness of the model and combine convolutional layers with an attention mechanism to avoid value redundancy in the attention mechanism. Finally, we propose a ProbSparse self-attentive mechanism combined with UniDrop to reduce the computational overhead and mitigate the risk of losing some key connections in the sequence. Experimental results on five datasets show that LDformer outperforms the state-of-the-art baseline for most of the results in the different long time series prediction tasks.

Key words: Long-term power forecasting; LDformer; LSTM; UniDrop; Self-attention mechanism
https://doi.org/10.1631/FITEE.2200540 **CLC number:**

1 Introduction

Power load forecasting is an important part of power system planning and the basis of the economic operation of power systems. Accurate power load forecasting research helps to provide a reliable decision-making basis for power system planning and operation, thus reducing power production costs (Ciechulski & Osowski, 2021). Traditional prediction methods have achieved great results in meteorology, finance, industry, and transportation in the short-term power load forecasting problem. However, existing time series methods still have limitations in long-time series prediction.

(1) Due to the highly complex and large scale of long-time series data, traditional methods are limited in efficiently handling high-dimensional large data and representing complex functions (Han et al., 2019). It will forget some data and ignore the long-term dependency of the time-series data.

(2) The model structure of traditional methods is not stable when performing long-term power load

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No.71961028, No.62261050), the Key Research and Development Program of Gansu (No.22YF7GA171), the Scientific Research Project of the Lanzhou Science and Technology Program (No.2018-01-58, No.2017-4-101), and the Natural Science Foundations of Gansu (No.21JR7RA119, No.21JR7RA208).

ORCID: Ran Tian, https://orcid.org/0000-0003-4435-580X;
Xinmei Li, https://orcid.org/0000-0002-5595-9230;
Zhongyu Ma, https://orcid.org/0000-0001-8809-0685;
Yanxing Liu, https://orcid.org/0000-0002-0554-3683;
Jingxia Wang, https://orcid.org/0000-0003-0696-9982;
Chu Wang, https://orcid.org/0000-0002-8687-9911

forecasting.

(3) The existing attention mechanism is prone to lose key connections between sequences when capturing the dependencies between long-time series data, leading to the degradation of prediction performance.

Therefore, aiming at the problem of long-time series prediction, we propose LDformer, a parallel neural network model for long-time series power load prediction based on the Informer framework (Zhou et al., 2020). The contributions of this model are summarized as follows:

(1) We propose a parallel neural network model LDformer based on Informer and LSTM to solve the long-time series load prediction problem. LSTM for feature extraction to obtain deep expression capability in time series.

(2) We propose a parallel encoder module that combines a convolutional layer and an attentional mechanism to avoid value redundancy in the attentional mechanism. Several experiments on different datasets validate the effectiveness and robustness of the parallel encoder and the convolutional layer.

(3) We propose a ProbSparse Self-attentive mechanism combined with UniDrop. UniDrop does not need additional computational overhead or external resources. Combining the UniDrop probabilistic sparse attention mechanism can mitigate the risk of losing some key connections in the sequence.

The rest of the paper is organized as follows. Section 2 presents the current state of research on time series forecasting and related models; Section 3 provides the problem definition for long time series power forecasting; Section 4 provides a detailed model introduction, expanding on each module in turn to explain; and Section 5 presents the experimental and analytical parts. Finally, the paper concludes in Section 6.

2 Related work

Long-time series prediction plays an essential role in decision-making in many fields, such as economy, transportation, medicine, hydrology, and energy (Chuku et al., 2019; Di et al., 2018; Chakraborty et al., 2019; Xie & Lou, 2019; Marcjasz et al., 2020). However, the prediction model may produce inaccurate results due to different patterns of the actual time series. In this paper, we analyse both machine learning and deep learning.

2.1 Machine Learning

In time series prediction, the ARIMA model is widely used in various fields. Chen et al. (1995) developed an adaptive ARIMA model for short-term power load forecasting of a power generation system. Viccione et al. (2019) applied the ARIMA model for tank water level prediction and analysis. Many researchers have also achieved better results by improving ARIMA models or combining ARIMA with other models. Xu et al. (2017) proposed an ARIMA and Kalman filter-based approach applied to road traffic real-time state prediction. Khan et al. (2020) proposed a wavelet-based hybrid ARIMA model. However, although the ARIMA model has achieved great success in stable time series forecasting, there are no almost pure stationary data in the real-time series data. Therefore, the application of the ARIMA model is limited by the data characteristics and is less general. As a result, many models other than ARIMA models have been applied to time series forecasting. Syriopoulos et al. (2020) used a support vector machine (SVM) to predict shipping prices. Based on SVM, Min et al. (2020) developed a time series model based on the least squares support vector machine (LSSVM) and achieved better results. Nobrega & Oliveira (2019) proposed sequential learning methods for the Kalman filter and a limited learning machine for regression and time series prediction, obtaining better results. In addition to the normal case, many researchers have also modelled predictions for time-series data with outliers. Chen & Sun (2021) then proposed a Bayesian time-factor decomposition framework for modelling and predicting multidimensional time series of specific spatiotemporal data in the presence of missing values. However, machine learning methods cannot obtain more accurate results for complex prediction problems.

2.2 Deep Learning

With the development of deep learning, researchers have found that deep learning applies to complex time series problems (Kim et al., 2018). Many scholars use convolutional neural networks (CNNs) to model and predict time series data (Guo et al., 2019; Hosseini & Talebpour, 2019). Jiasheng &

Jinghan (2019) applied CNN to stock forecasting. However, a convolutional neural network is more suitable for spatial correlation than time series. As a result, recurrent neural networks have emerged. Hu et al. (2020) applied recurrent neural networks (RNNs) to traffic flow prediction based on time series analysis. Min et al. (2019) proposed an RNN-based intent inference model to solve the time series prediction problem. Many studies have proven that RNNs have obvious advantages in time series prediction (Zhang et al., 2019; Xiao et al., 2019). However, when the time series is too long, the problems of gradient disappearance and gradient explosion may occur in RNN training (Zheng & Huang, 2020).

Long- and short-term memory networks (LSTM) (Karevan & Suykens, 2020) were proposed to solve these problems. LSTM is more suitable for long time series than RNN. Therefore, many deep learning models based on LSTM are applied to time series prediction. Zhang et al. (2019) used LSTM for gas concentration prediction. Kcm et al. (2020) proposed a novel LSTM framework for short-term fog prediction, consisting of an LSTM and a fully connected layer. Their experiments showed that the framework outperformed KNN, AdaBoost, and CNN algorithms. Gai et al. (2021) proposed a new parking space prediction model based on LSTM, which provides a more accurate prediction. Many researchers have also proposed a combined model of LSTM and other models (Ran et al., 2019; Wang et al., 2020), which proves the feasibility of LSTM in time series prediction. However, through the literature (Khandelwal et al., 2018), we can learn that the adequate context size of the language model using LSTM is approximately 200 tokens on average. Nevertheless, it can only distinguish 50 tokens in the vicinity, which indicates that even LSTM has difficulty capturing long-term dependencies.

In recent years, the transformer has been applied to many fields to perform long-time series prediction tasks. Neo Wu et al. (2020) applied it to the prediction of influenza-like diseases, and this method can learn complex patterns and dynamics from time-series data by using the self-attention mechanism. As a general framework, the transformer can not only be applied to univariate and multivariate time series data but can also be used for time series embedding. However, because the point-by-point dot product in the transformer architecture is insensitive to the local context, its spatial complexity is too large. Therefore, Li et al. (2019) proposed the logspare transformer, which improves the prediction accuracy of time series with fine granularity and strong long-term dependence under a limited memory budget. In addition, Zhou et al. (2020) proposed the Informer and applied it to electricity consumption planning. Unlike the self-attention mechanism used in the transformer (Vaswani et al., 2017), this model proposes a new ProbSparse Self-attentive (Zhou et al., 2020), which minimizes the complexity and improves the transformer model's calculation, memory, and algorithm efficiency. Although the existing time series prediction methods have promoted the development of the time series field to a certain extent, there is still room for improvement. Because of the increase in data volume, they ignore the deep expression ability of temporal data. They have problems such as many parameters, memory consumption, and slow running time.

Compared with previous work, LDformer not only considers the deep representation of temporal data but also mitigates the risk of losing critical connections between sequences while ensuring minimum complexity. In addition, LDformer combines the convolutional layer and the attention mechanism to obtain a parallel encoder module, which prevents redundancy in the attention mechanism while improving model robustness. These methods effectively improve prediction accuracy.

3 Preliminary

In the prediction setting with a fixed length, we have input $\mathbf{X}^t=\left\{\mathbf{x}_1^t,...,\mathbf{x}_{Lx}^t\middle|\mathbf{x}_i^t\in R^{dx}\right\}$ at time *t*, and the output is to predict the corresponding sequence $\mathbf{Y}^t=\left\{\mathrm{y}_1^t,...,\mathrm{y}_{Ly}^t\middle|\mathrm{y}_i^t\in R^{dy}\right\}$. This paper is multivariate predicting univariate. where Lx is the number of input features. Ly is the number of output sequences. The goal of power long time series forecasting is to map the power load profile at *m* time steps in the history to *n* time steps in the target feature.

$$[\mathbf{X}_{t-m+1},\mathbf{X}_{t-m+2},...,\mathbf{X}_t]\xrightarrow{f(\cdot)}[\mathrm{y}_{t+1},\mathrm{y}_{t+1},...,\mathrm{y}_{t+n}] \quad (1)$$

4 Long time series prediction model

In this section, the overall framework of the LDformer is shown in Fig. 1.

LDformer contains LSTM, encoder and decoder structures. First, the time-series data enter the embedding layer, which considers data, location, and time information converted into a uniform dimension and then merged. The data processed by the embedding layer enter the LSTM, which can use the long-range temporal information more effectively. Second, the data enter the encoder, which uses a multichannel parallel mode to improve model robustness to receive long sequence inputs. At the same time, the convolution layer is added between encoder modules to reduce the redundant combination of encoder feature mapping with value. The decoder receives long sequence inputs and predicts the output element immediately in the form of generation. After the embedding layer, the data enter the decoder. To ensure that the output decoded at time t depends only on the output before time t, $\mathbf{X}_0$ is a placeholder for the target sequence, padded with 0. A mask is added to the first attention of the decoder to prevent the target information from being used earlier. Finally, the prediction results of the last column are output through a fully connected layer.

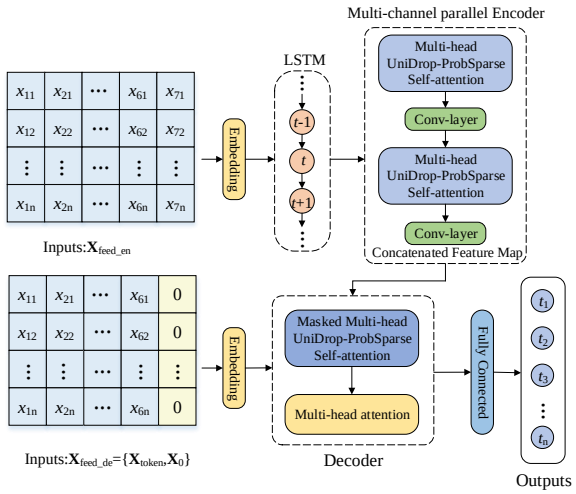


Fig. 1: The framework of Ldformer

4.1 Embedding layers with multiple perspectives

First, the preprocessed data are unified into dimensions, and data coding, location coding and timestamp coding are performed. Then, the final embedding result is obtained by summing the three types of coding.

(1) Data Embedding (DE): Convert data dimension x-in to uniform dimension $\mathrm{d}_{\mathrm{model}}$ using one-dimensional convolution. The formula is as follows:

$$\mathrm{DE}=\mathrm{conv1d}(x\text{-in},\mathrm{d}_{\mathrm{model}}) \tag{2}$$

(2) Position Embedding (PE) (Vaswani et al., 2017): The elements in the input sequence are processed together, which differs from RNN processing. Although the RNN's processing speed is faster than that of PE, it ignores the sequence of elements in the sequence, so we choose positional embedding. The formulas are as follows:

$$\mathrm{PE}(\mathrm{pos},2i)=\sin\left(\mathrm{pos}\Big/10000^{\frac{2i}{\mathrm{d}_{\mathrm{model}}}}\right) \tag{3}$$

$$\mathrm{PE}(\mathrm{pos},2i+1)=\cos\left(\mathrm{pos}\Big/10000^{\frac{2i}{\mathrm{d}_{\mathrm{model}}}}\right) \tag{4}$$

where pos denotes the position of the element in the sequence; $\mathrm{d}_{\mathrm{model}}$ denotes the dimension of the element vector; and i denotes the position of the element vector.

(3) Time Embedding: There are various methods for time embedding: month_embed, day_embed, weekday_embed, hour_embed, and minute_embed. The time slice of the dataset used in this paper is mainly in minutes and hours, so hour_embed and minute_embed are chosen to obtain the time stamp encoding results.

Summing these three embedding components is the output of the final embedding layer. The overall embedding layer structure is shown in Fig. 2.

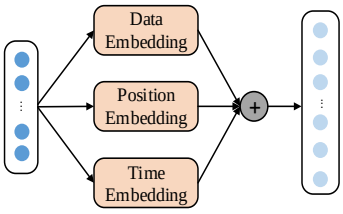


Fig. 2: Embedding

The data output by embedding are sent to LSTM for feature extraction. To not affect the input of the subsequent encoder, this paper makes the output of LSTM consistent with the output of Embedding, i.e.,

the LSTM output dimension remains as d_{model}. The output of the LSTM is the input of the encoder. LSTM can perform better in long sequences. Therefore, using LSTM can extract the deep expression ability in the time series and improve prediction accuracy.

4.2 Multichannel parallel encoder module

We build the encoder module by combining the attention mechanism with the convolutional layer. It takes four channels of length L, L/2, L/4, and L/8 and executes them in parallel. The convolutional layer performs dimensional pruning and reduces the memory footprint before the output of the upper layer is sent to the lower layer of the multiheaded attention module. The convolutional layer is always one less layer than the encoder layer. The multichannel parallel encoder module is shown in Fig. 3.

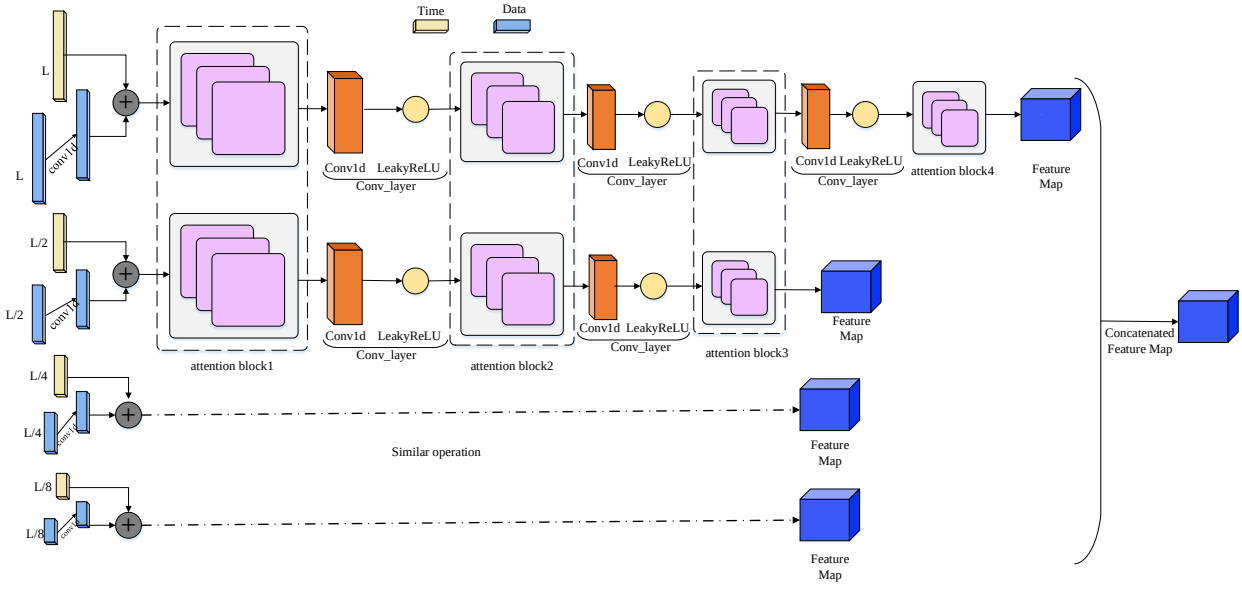


Fig. 3: Multichannel parallel encoder module

The encoder is mainly used to extract robust remote dependencies from time series data. The overall architecture of the encoder here is roughly the same as that of the transformer. It mainly includes two sublayers, the multihead attention layer (ProbSparse Self-attentive mechanism combined with UniDrop) and the feed-forwards layer composed of two linear mappings. A batch normalization layer follows both sublayers, with jump connections between the sublayers.

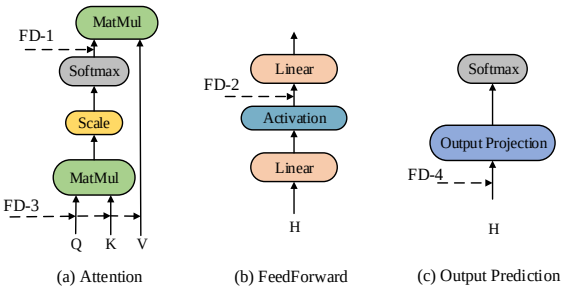
4.2.1 ProbSparse Self-attentive mechanism combined with UniDrop

The ProbSparse self-attention mechanism may create some risk of critical connection loss. To address this problem, we propose a ProbSparse attention mechanism combined with UniDrop. Unlike the self-attention mechanism and the probabilistic sparse self-attention mechanism, we consider other possible problems while retaining their advantages. The canonical self-attention mechanism consists of a query and a set of key-value pairs. The formula is as follows (Vaswani et al., 2017).

$$\mathbf{A}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V} \tag{5}$$

where $\mathbf{Q}\in R^{L_\mathbf{Q}\times d}$, $\mathbf{K}\in R^{L_\mathbf{K}\times d}$, $\mathbf{V}\in R^{L_\mathbf{V}\times d}$ and d is the input dimension. This method has high time complexity. Therefore, Zhou et al. (2020) proposed ProbSparse self-attention. However, due to the high number of parameters in the attention mechanism and the tendency of overfitting and loss of key connections between sequences, we introduce the UniDrop technique (Wu et al., 2021), in which feature dropout (FD) can randomly suppress some neurons in the network with a certain probability. FD-1 is applied to attention weight $\mathbf{A}$, which is used to improve the generalization of multihead attention. FD-2 is applied after the activation function between the two linear transformations of the feed-forwards network sublayer.

However, it is directly used to the attention weights **A**, where a drop value A(*i*, *j*) means ignoring the relationship between token *i* and token *j*. Thus, a larger FD-1 means a larger risk of losing some critical information from sequence positions. We add dropout to **Q**, **K**, and **V** to alleviate this potential risk before calculating attention. FD-4 is used for the output features before the linear transformation. The overall structure is shown in Fig. 4.



(a) Attention (b) FeedForward (c) Output Prediction

Fig. 4: UniDrop

Define the attention of the *i*-th row, *i*-th $\mathbf{q}^{'}$ of $\mathbf{Q}^{'},\mathbf{K}^{'},\mathbf{V}^{'}$ obtained after dropout as a kernel smoother in the form of probability.

$$\mathbf{A}\left(\mathbf{q}_i^{'},\mathbf{K}^{'},\mathbf{V}^{'}\right)=\sum_j\frac{k\left(\mathbf{q}_i^{'},\mathbf{k}_j^{'}\right)}{\sum_l k\left(\mathbf{q}_i^{'},\mathbf{k}_l^{'}\right)}\mathbf{v}_j^{'}=E_{p(\mathbf{k}_j|\mathbf{q}_i^{'})}\left[\mathbf{v}_j^{'}\right] \quad (6)$$

In formula (5), the attention of the *i*-th query on all keys is defined as the probability $p\left(\mathbf{k}_j^{'}\middle|\mathbf{q}_i^{'}\right)$, and the formula is as follows.

$$p\left(\mathbf{k}_j^{'}\middle|\mathbf{q}_i^{'}\right)=\frac{k\left(\mathbf{q}_i^{'},\mathbf{k}_j^{'}\right)}{\sum_l k\left(\mathbf{q}_i^{'},\mathbf{k}_l^{'}\right)} \quad (7)$$

where $p\left(\mathbf{k}_j^{'}\middle|\mathbf{q}_i^{'}\right)$ and $k\left(\mathbf{q}_i^{'},\mathbf{k}_j^{'}\right)$ select the asymmetric exponential kernel $\exp\left(\frac{\mathbf{q}_i^{'}\mathbf{k}_j^{'\mathrm{T}}}{\sqrt{d}}\right)$. Self-attention combines these values and the output obtained based on $p\left(\mathbf{k}_j^{'}\middle|\mathbf{q}_i^{'}\right)$. However, to avoid the near-uniform distribution $q\left(\mathbf{k}_j\middle|\mathbf{q}_i\right)=\frac{1}{L_{\mathrm{k}}}$ of $p\left(\mathbf{k}_j^{'}\middle|\mathbf{q}_i^{'}\right)$ so that the attention values become the sum of $\mathbf{V}^{'}$, resulting in input redundancy, we use Kullback–Leibler (KL) dispersion to select important queries through the similarity distribution between p and q (Zhou et al., 2020). The formula is as follows:

$$\mathrm{KL}\left(q\|p\right)=\ln\sum_{l=1}^{L_{\mathbf{K}^{'}}}e^{\frac{\mathbf{q}_i\mathbf{k}_l^{'\mathrm{T}}}{\sqrt{d}}}-\frac{1}{L_{\mathbf{K}^{'}}}\sum_{j=1}^{L_{\mathbf{K}^{'}}}\frac{\mathbf{q}_i^{'}\mathbf{k}_j^{'\mathrm{T}}}{\sqrt{d}}-\ln L_{\mathbf{K}^{'}} \quad (8)$$

Dropping the constant, the sparsity measure of the *i*-th query can be defined as:

$$\mathrm{M}\left(\mathbf{q}_i^{'},\mathbf{K}^{'}\right)=\ln\sum_{j=1}^{L_{\mathbf{K}^{'}}}e^{\frac{\mathbf{q}_i^{'}\mathbf{k}_j^{'\mathrm{T}}}{\sqrt{d}}}-\frac{1}{L_{\mathbf{K}^{'}}}\sum_{j=1}^{L_{\mathbf{K}^{'}}}\frac{\mathbf{q}_i^{'}\mathbf{k}_j^{'\mathrm{T}}}{\sqrt{d}} \quad (9)$$

However, traversing all measurement queries requires computing each dot product pair, and the LSE operation has potential numerical stability problems. Based on this, the above formula is again improved to obtain the final sparsity measure formula.

$$\mathrm{M}\left(\mathbf{q}_i^{'},\mathbf{K}^{'}\right)=\max_j\left\{\frac{\mathbf{q}_i^{'}\mathbf{k}_j^{'\mathrm{T}}}{\sqrt{d}}\right\}-\frac{1}{L_{\mathbf{K}^{'}}}\sum_{j=1}^{L_{\mathbf{k}^{'}}}\left\{\frac{\mathbf{q}_i^{'}\mathbf{k}_j^{'\mathrm{T}}}{\sqrt{d}}\right\} \quad (10)$$

Based on the above steps, we obtain ProbSparse self-attention combined with UniDrop by allowing each key to pay attention to only *u* dominant queries.

$$\mathbf{A}\left(\mathbf{Q}^{'},\mathbf{K}^{'},\mathbf{V}^{'}\right)=\mathrm{Softmax}\left(\frac{\overline{\mathbf{Q}}^{'}\mathbf{K}^{'T}}{\sqrt{d}}\right)\mathbf{V}^{'} \quad (11)$$

where $\overline{\mathbf{Q}}^{'}$ is a sparse matrix of the same size as $\mathbf{q}^{'}$ containing only Top-*u* queries under the sparse metric, taking the part of them with higher probability. We set $u=c\cdot\ln L_{\mathbf{Q}^{'}}$ and controlled by a constant sampling factor *c*. The algorithmic procedure for ProbSparse self-attention combined with UniDrop is shown in Algorithm 1.

Algorithm 1 ProbSparse Self-attentive mechanism combined with UniDrop

Input: some thing

Input: Tensor $\mathbf{Q}\in R^{m\times d}$, $\mathbf{K}\in R^{n\times d}$, $\mathbf{V}\in R^{n\times d}$

1 Initialize: set hyperparameter c, $u=c\ln m$ and $U=m\ln n$, Dropout parameter p

2 $\mathbf{Q}^{'}=\mathrm{dropout}(\mathbf{Q}),\mathbf{K}^{'}=\mathrm{dropout}(\mathbf{K}),\mathbf{V}^{'}=\mathrm{dropout}(\mathbf{V})$

3 arbitrarily select U dot-product pairs from $\mathbf{K}^{'}$ as $\overline{\mathbf{K}}^{'}$

4 Sample $\mathbf{K}^{'}$ and set the sample score $\overline{\mathbf{S}}=\mathbf{Q}^{'}\overline{\mathbf{K}}^{'T}$

5 each $\mathbf{q}_i$ on K_sample Calculate M value

6 Set top-u queries as $\overline{\mathbf{Q}}^{'}$ based on formula M.

7 set $\mathbf{A}_0=\mathrm{softmax}\left(\overline{\mathbf{Q}}^{'}\mathbf{K}^{'T}/\sqrt{d}\right)\cdot\mathbf{V}^{'}$

8 for the value of the score of unchecked $\mathbf{q}_i^{'}$ set $\mathbf{A}_1=\mathrm{mean}\left(\mathbf{V}^{'}\right)$

9 set self-attention formula $\mathbf{A}=\left\{\mathbf{A}_0,\mathbf{A}_1\right\}$

Output: self-attention feature map **A**

4.2.2 Convolutional layer

As a natural consequence of the ProbSparse attention mechanism combined with UniDrop, the feature mapping of the encoder produces a redundant combination of values $\mathbf{V}^{'}$. Therefore, in the next layer, we use convolution to extract dominant features for processing so that they generate a focused attentional feature map, as shown in Fig. 3, which will largely reduce the temporal dimension of the input. CNN is very good at identifying simple patterns in data and generating complex patterns in more advanced layers. Conv1D is very effective for obtaining features of interest from data whose locations are not highly correlated, and Conv1D can be well applied to time series analysis of sensor data. Therefore, we select Conv1D to extract features and set its convolution kernel as 3x3. The formula is as follows.

$$\mathbf{X}_{j+1}^{t}=\text{LeakyReLU}\left(\text{Conv1d}\left(\left[\mathbf{X}_{j}^{t}\right]_{AB}\right)\right) \quad (12)$$

where $\left[\cdot\right]_{AB}$ contains the basic operations in the multiheaded attention and attention block, and Conv1D uses the Leaky-ReLU activation function to execute in the time dimension. LeakyReLU is a variant of ReLU, which changes the response to the input less than 0 part, alleviates the sparsity of ReLU, and inherits the advantages of ReLU, which can speed up the convergence, alleviate the gradient disappearance and explosion problems, and simplify the calculation. The LeakyReLU activation function formula is shown as follows.

$$\text{LeakyReLU}(x)=\max(0,x)+\text{negative\_slope}\cdot\min(0,x) \quad (13)$$

4.3 Decoder

The decoder generates the time series output by a forwards process, and part of its structure can be referred to as the decoder structure in the transformer, as shown in Fig. 5.

The decoder consists of two layers of attention mechanisms and a linear mapping feed-forwards layer. Its input vector is as follows:

$$\mathbf{X}_{feed\_de}^{t}=Concat\left(\mathbf{X}_{token}^{t},\mathbf{X}_{0}^{t}\right)\in R^{\left(L_{token}+L_{y}\right)\times d_{\text{model}}} \quad (14)$$

where $\mathbf{X}_{token}^{t}\in R^{L_{token}\times d_{\text{model}}}$ is the start token and $\mathbf{X}_{0}^{t}\in R^{L_{y}\times d_{\text{model}}}$ is a placeholder for the target sequence. The first layer of attention is the probabilistic sparse self-attention of the combined UniDrop, as shown in formula (11). We set the masked multi-headed self-attention to $-\infty$. It prevents focusing on the future position in the training process to avoid the autoregression problem. The second layer of attention is ordinary self-attention, as shown in formula (5). An Add&Norm layer follows both layers of attention. Add&Norm is composed of two parts, Add and Norm, and is calculated as follows.

$$\mathbf{X}=\text{LayerNorm}\left(\mathbf{X}+\text{MultiHeadAttention}\left(\mathbf{X}\right)\right) \quad (15)$$

Finally, the prediction results are output directly through a fully connected layer.

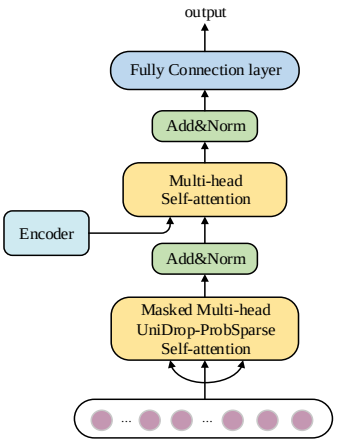


Fig. 5: Decoder for growing sequence output through forwards process generation

5 Experiment and Analysis

5.1 Datasets

We extensively perform experiments on five datasets, namely, the ETTm1, ETTh1, ETTh2, PEMS03 and weather datasets. ETTm1, ETTh1, and ETTh2 are collectively referred to as the ETT dataset. Table 1 shows the description of the datasets.

ETT (Zhou et al., 2020): ETT is a key indicator of the long-term deployment of electricity. Here are two years of data collected from two different counties in China. ETTm1 takes one data point every 15 minutes. ETTh1 and ETTh2 take one data point every hour. Each data point consists of the target value "OT" and six different types of external load values: "High UseFul Load (HUFL)", "High UseLess Load (HULL)", "Middle UseFul Load (MUFL)", "Middle UseLess Load (MULL)", "Low UseFul Load (LUFL)", and "Low UseLess Load (LULL)". We use a multivariate prediction univariate model with six power load features to predict the target value "Oil Temperature (OT)." According to the time characteristics, we divide the training set, validation set and

test set by 12/4/4 months.

Weather (Zhou et al., 2020): This dataset contains local climate data for nearly 1600 locations in the United States. Each data point consists of a target value "wet bulb" and 11 climate features. We divide the training set, validation set and test set by 12/4/4 months.

PEMS03(Wang et al., 2023): This dataset is the traffic flow data of the California highway network. It is divided into five-minute intervals and contains data from 307 sensors for three months from 2018/9/1 to 2018/11/30. This experiment intercepts the traffic data of the first seven sensors for three months. The dataset is divided by 7:2:1 into a training set, test set, and validation set. The seventh sensor is used as the target sensor for the experiment.

Table 1 Datasets and prediction task descriptions

Datasets	Data Description	Time range	Time interval	Target Features	Target length	Length of time
ETTm1	Key indicators for long-term power deployment collected from two different counties in China.	07/01/2018 -06/26/2018	15 min	"OT"	{24, 36, 48, 96, 168, 336}	{6 h, 9 h, 12 h, 24 h, 42 h, 84 h}
ETTh1		07/01/2018 -06/26/2018	1 h	"OT"	{24, 36, 48, 96, 168, 336}	{24 h, 36 h, 48 h, 96 h, 168 h, 336 h}
ETTh2		07/01/2018 -06/26/2018	1 h	"OT"	{24, 36, 48, 96, 168, 336}	{24 h, 36 h, 48 h, 96 h, 168 h, 336 h}
Weather	Local climate data for nearly 1,600 U.S. locations.	01/01/2010 -12/31/2013	1 h	"wet bulb"	{24, 48, 96, 168, 336, 720}	{24 h, 48 h, 96 h, 168 h, 336 h, 720 h}
PEMS03	Traffic flow dataset	09/01/2018 -11/30/2018	5 min	"313512"	{24, 48, 96, 228, 336, 720}	{2 h, 4 h, 8 h, 24 h, 28 h, 60 h}

5.2 Experimental Setting

In this paper, the model uses both parallel and nonparallel modes in the encoder. The parallel mode has four channels executing in parallel and includes three stacks. The decoder contains two stacks. The sampling factor c in the Top-u formula is set to 5. The parameter in dropout is 0.1. The number of multihead attention *n*-heads is set to 8. When predicting the target sequence, MSE is selected as the loss function, and the whole model is returned from the output of the decoder. The learning rate of the experimental setup decreases from $1e^{-4}$ and decays by a factor of 2 for each period. Optimization with Adam Optimizer. Set the number of epochs equal to 10 to stop early when good results are obtained, and the batch size is 64 and standardized each input dataset before the experiment.

Assessment metrics: This experiment uses three evaluation indexes: MSE, MAE, and RMSE. MSE and MAE are the mean square error and mean absolute error, respectively. RMSE is the root mean square error. The main reason for choosing RMSE is that it will be more intuitive in terms of order of magnitude compared to MSE. For example, when the RMSE is equal to 10, then the regression effect can be considered to differ by ten on average compared to the true value. The indicator equation is as follows.

$$\mathrm{MSE}=\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{y}-\hat{\mathbf{y}}\right)^{2} \tag{16}$$

$$\mathrm{MAE}=\frac{1}{n}\sum_{i=1}^{n}\left|\mathbf{y}-\hat{\mathbf{y}}\right| \tag{17}$$

$$\mathrm{RMSE}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{y}-\hat{\mathbf{y}}\right)^{2}} \tag{18}$$

where $\mathbf{y}$ is the true value and $\hat{\mathbf{y}}$ is the predicted value.

5.3 Experimental Results and Analysis

We have selected recurrent neural networks RNN, LSTM, and GRU, often used for long-time series prediction, and the representative algorithm informer and its informer (NP) model under nonparallel conditions in recent years. We compare them under the same conditions with our parallel model LDformer and nonparallel LDformer (NP) on five datasets. The specific description of the dataset is shown in Table 1. The corresponding experimental results are shown in Table 2, Table 3, Table 4 and Table 5.

Table 2 and Table 3 show the prediction performance of the models in the baseline and this paper from short to long series of the ETT dataset, respectively. As the prediction length increases, the prediction performance of LDformer continues to improve in all datasets. This demonstrates the success of LDformer in improving prediction performance in

long-time series prediction problems. In the ETTm1 dataset, when the prediction length is 48, the indexes MSE, MAE, and RMSE of the LDformer model proposed in this paper are 36.9%, 22.4%, and 20.6% higher than those of the Informer. LDformer (np) is also better than Informer. As the length of the data increases, the advantages of the LDformer become more apparent. With a prediction length of 168, LDformer compared to Informer metrics MSE, MAE, and RMSE improved by 16.4%, 13.1%, and 8.6%, respectively. The optimal and suboptimal results are obtained for the models LDformer (np) and LDformer in this paper for a prediction length of 336.

In both the ETTh1 and ETTh2 datasets, the proposed model outperforms the conventional model for short-series prediction. Taking the prediction length in ETTh1 equal to 24, our proposed model's LDformer indexes MSE, MAE, and RMSE are 74.6%, 44.3%, and 49.6% higher than those of the traditional GRU model, respectively. After increasing the prediction length, Informer shows good performance in the ETTh1 dataset when the prediction length is 96,168. However, LDformer is better than the traditional model in most cases. In the ETTh2 dataset long-time series prediction, the model in this paper is inferior to the traditional model. First, this phenomenon may be caused by the anisotropy of the feature dimensions. It is beyond the scope of this paper, and we will explore it in future work. Second, ETTh2 contains a large amount of continuous null data.

Table 2. Performance comparison of short time series prediction tasks on the ETT dataset

(The best results are in bold, and * denotes the second-best results)

Predict length		pred=24			pred=36			pred=48		
Metrics		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
ETTm1	RNN	5.0206	1.9853	2.2407	2.7855	1.4088	1.6689	5.7508	2.1783	2.3981
	LSTM	2.6396	1.3441	1.6247	2.7410	1.3751	1.6555	2.7252	1.3704	1.6508
	GRU	1.9783	1.1282	1.4065	5.0422	1.9861	2.1454	3.0284	1.4052	1.7402
	Informer(np)	0.0904	0.2279	0.3006	0.0816	0.2247	0.2852	0.0739*	0.2117*	0.2719*
	Informer	**0.0432**	**0.1606**	**0.2079**	**0.0674**	**0.2021**	**0.2597**	0.1055	0.2571	0.3248
	LDformer(np)	0.0605*	0.1962*	0.2458*	0.0800	0.2271	0.2830	0.0999	0.2574	0.3161
	LDformer	0.0789	0.2301	0.2810	0.0754*	0.2136*	0.2747*	**0.0665**	**0.1993**	**0.2578**
ETTh1	RNN	3.6319	1.6149	1.9057	4.3593	1.8123	2.0879	4.6225	1.8864	2.1500
	LSTM	2.1395	1.1835	1.4627	2.7266	1.3697	1.6512	2.5198	1.3068	1.5874
	GRU	0.9840	0.7832	0.9920	0.9556	0.7625	0.9775	1.9024	1.1236	1.3792
	Informer(np)	0.3420*	0.5325*	0.5848*	0.3861*	0.5532*	0.6314*	0.5013	0.6441	0.7080
	Informer	0.4798	0.6475	0.6926	0.5079	0.6685	0.7127	0.3964*	**0.5406**	0.6296*
	LDformer(np)	0.4193	0.5937	0.6475	**0.3458**	**0.5314**	**0.5880**	0.5183	0.6545	0.7199
	LDformer	**0.2492**	**0.4361**	**0.4992**	0.4304	0.5859	0.6560	**0.3782**	0.5579*	**0.6150**
ETTh2	RNN	2.2385	1.2577	1.4961	2.0350	1.1824	1.4265	1.4774	1.0111	1.2155
	LSTM	1.0688	0.8114	1.0338	1.1154	0.8529	1.0561	0.6622*	0.6543*	0.8138*
	GRU	0.7586*	0.7136*	0.8710*	0.7955	0.7204*	0.8919	1.0504	0.8336	1.0249
	Informer(np)	1.0555	0.9186	1.0273	1.2231	1.0052	1.1059	1.7355	1.2214	1.3173
	Informer	1.2559	1.0338	1.1206	1.1121	0.9529	1.0545	2.0611	1.3422	1.4356
	LDformer(np)	0.7810	0.7721	0.8837	**0.5988**	**0.6642**	**0.7738**	0.7493	0.9541	1.0720
	LDformer	**0.5713**	**0.6384**	**0.7558**	0.7553*	0.7575	0.8691*	**0.7224**	**0.7296**	**0.8499**

Table 3. Performance comparison of long time series prediction tasks on the ETT dataset

(The best results are in bold, and * denotes the second-best results)

Predict length		pred=96			pred=168			pred=336		
Metrics		MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
ETTm1	RNN	3.7047	1.6129	1.9247	4.9325	1.9187	2.2209	3.4728	1.5242	1.8635
	LSTM	2.7349	1.3728	1.6537	2.7401	1.3749	1.6553	2.7471	1.3763	1.6574
	GRU	3.6131	1.5948	1.9008	2.8525	1.4074	1.6889	3.1644	1.4829	1.7788
	Informer(np)	0.3387*	0.5361	0.5820*	0.6584	0.7109	0.8114	1.0142	0.9140	1.0070

	Informer	0.5603	0.6874	0.7485	0.6100	0.7097	0.7810	1.0266	0.9412	1.0132
	LDformer(np)	**0.3248**	**0.5142**	**0.5699**	0.5146*	0.6573*	0.7173*	**0.5357**	**0.6519**	**0.7319**
	LDformer	0.3523	0.5277*	0.5936	**0.5096**	**0.6166**	**0.7138**	0.6346*	0.7103*	0.7966*
ETTh1	RNN	3.5576	1.5451	1.8861	3.4802	1.5640	1.8655	2.8218	1.3619	1.6798
	LSTM	2.2462	1.2268	1.4987	2.2827	1.2244	1.5108	2.0860	1.1430	1.4443
	GRU	2.6474	1.3466	1.6271	1.8953	1.0810	1.3767	2.3861	1.2097	1.5447
	Informer(np)	**0.2577**	**0.4236**	**0.5076**	0.3024*	0.4721*	0.5499*	0.7343	0.7743	0.8569
	Informer	0.3014*	0.4639*	0.5490*	**0.2656**	**0.4413**	**0.5154**	0.9416	0.8984	0.9703
	LDformer(np)	0.4545	0.6069	0.6741	0.4046	0.5617	0.5615	0.5057*	**0.6421**	0.7111*
	LDformer	0.3106	0.4880	0.5569	0.4886	0.6322	0.5224	**0.5034**	0.6455*	**0.7095**
ETTh2	RNN	1.5415	1.0042	1.2416	2.0284	1.1726	1.4242	2.1122	1.1998	1.4533
	LSTM	**1.0823**	**0.8412**	**1.0403**	1.1039*	0.8520*	1.0507*	**1.1152**	0.8568*	**1.0560**
	GRU	1.1424*	0.8764*	1.0688*	**0.8537**	**0.7320**	**0.9239**	1.1582*	**0.8390***	1.0762*
	Informer(np)	2.2963	1.4216	1.5153	2.3733	1.4409	1.5405	2.2128	1.3732	1.4875
	Informer	2.7244	1.5558	1.6506	2.9010	1.6124	1.7032	2.1025	1.3379	1.4500
	LDformer(np)	2.1959	1.3798	1.4818	2.2004	1.3807	1.4834	2.8714	1.5847	1.6945
	LDformer	1.8590	1.2490	1.3634	2.5229	1.4819	1.5883	2.6768	1.5261	1.6361

Table 4. Performance comparison of short time series prediction tasks on the Weather and PEMS03 datasets (The best results are in bold, and * denotes the second-best results)

Predict length		pred=24			pred=48			pred=96		
	Metrics	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Weather	RNN	0.9135	0.5446	0.9558	1.0542	0.6193	1.0267	1.5387	0.8906	1.2404
	LSTM	0.6073	0.4037	0.7793	0.6203	0.5168	0.7875	1.0249	0.5648	1.0123
	GRU	0.4992	0.3378	0.7066	0.5184	0.5661	0.7200	0.7345	0.6472	0.7965
	Informer(np)	0.1666	0.2862	0.4082	0.3414*	0.4285*	0.5843*	0.7039	0.6236	0.8390
	Informer	0.1594*	0.2809*	0.3993*	0.3802	0.4552	0.6166	0.6459	0.5981	0.8037
	LDformer(np)	0.1650	0.2905	0.4062	**0.3393**	**0.4279**	**0.5825**	**0.5638**	**0.5594**	**0.7508**
	LDformer	**0.1574**	**0.2918**	**0.3968**	0.3677	0.4484	0.6064	0.5653*	0.5672*	0.7519*
PEMS03	RNN	0.4110	0.5089	0.6411	0.1498	0.3166	0.3871	0.8322	0.7091	0.9122
	LSTM	0.0776	0.2191	0.2786	0.0835*	0.2291	0.3089	0.2132	0.2644	0.4364
	GRU	0.0779	0.2206	0.2791	0.0842	0.2309	0.3002	0.2098	0.3635	0.4313
	Informer(np)	0.0582	0.1772	0.2413	0.0970	0.2158	0.3115	**0.1145**	**0.2377**	**0.3384**
	Informer	0.0553*	0.1705*	0.2352*	0.0891	0.2070	0.2986*	0.1195*	0.2426*	0.3457*
	LDformer(np)	**0.0551**	**0.1688**	**0.2348**	0.0915	**0.2054**	0.3024	0.1507	0.2568	0.3882
	LDformer	0.0650	0.1788	0.2550	**0.0865**	0.2055*	**0.2942**	0.1279	0.2400	0.3576

Table 5. Performance comparison of long time series prediction tasks on the Weather and PEMS03 datasets (The best results are in bold, and * denotes the second-best results)

Predict length		pred=168			pred=336			pred=720		
	Metrics	MSE	MAE	RMSE	MSE	MAE	RMSE	MSE	MAE	RMSE
Weather	RNN	1.6785	1.0261	1.2955	1.9695	1.1659	1.4034	1.9957	1.1651	1.4126
	LSTM	0.9975	0.6133*	0.9987	1.0505	0.7713	1.0249	1.0395	0.7682	1.0195
	GRU	0.7594	0.6360	0.8714	0.8735	0.6890	0.9346	1.6688	0.9437	1.2918
	Informer(np)	0.8772	0.7056	0.9366	0.8803	0.7128	0.9382	0.8857	0.7244	0.9411
	Informer	0.7835	0.6681	0.8851	0.8791	0.7196	0.9376	0.8743	0.7194	0.9350
	LDformer(np)	0.6598*	0.6134	0.8122*	**0.7277**	0.6485*	**0.8531**	**0.7751**	**0.6699**	**0.8804**
	LDformer	**0.6194**	**0.6040**	**0.7870**	0.7452*	**0.6477**	0.8632*	0.7881*	0.6807*	0.8877*
	Predict length		pred=288			pred=336			pred=720	
S03	RNN	1.1370	0.7934	1.0663	1.0338	0.7569	1.0167	1.3234	0.8594	1.1504

LSTM	0.8874	0.8013	0.9420	1.0287	0.8595	1.0142	0.3207	0.4365	0.5663
GRU	0.2669	0.4190	0.5085	0.2323	0.3918	0.4637	0.2395	0.4963	0.5735
Informer(np)	0.2264	0.3100	0.4758	0.1917*	0.2941	0.4378*	0.2252	0.3112	0.4746
Informer	0.2039	0.2967	0.4515	0.2090	0.3091	0.4571	0.2264	0.3200	0.4758
LDformer(np)	0.1759*	0.2690*	0.4195*	0.1957	0.2885*	0.4424	**0.2184**	**0.3046**	**0.4674**
LDformer	**0.1603**	**0.2617**	**0.4004**	**0.1826**	**0.2788**	**0.4273**	0.2251*	0.3087*	0.4744*

Table 4 and Table 5 show the prediction performance of the baseline model and this paper's model for short and long series in the Weather and PEMS03 datasets. As the prediction length increases, LDformer achieves better performance. In the weather dataset, when the prediction length is 168, the MSE, MAE and RMSE of LDformer are improved by 29.3%, 14.3%, and 15.9%, respectively, compared to Informer(np). In the PEMS03 dataset, when the prediction length is 336, the MSE, MAE and RMSE of LDformer are improved by 12.6%, 9.8% and 6.5%, respectively, compared to Informer. The overall experiments show that the model in this paper achieves better results in the PEMS03 dataset with a time interval of 5 minutes and the ETTm1 dataset with a time interval of 15 minutes.

LDformer accurately obtains the temporal information of each feature, where LSTM learns the long-term dependency of temporal data and fully exploits the depth expression ability among time series data. The probabilistic sparse attention mechanism combined with UniDrop inherits the advantages of the ProbSparse attention mechanism on the basis of avoiding the attention mechanism losing some key connections in the sequence when considering data correlation. Meanwhile, the encoder module adopts a multichannel parallel model to improve the robustness of the whole model.

We calculated the average error between the true and predicted values in different datasets. The error is plotted as a histogram, as shown in Fig. 6 and Fig. 7.

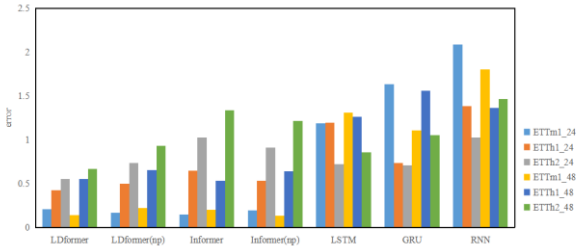


Fig. 6: Average error of the true and predicted values of the short series

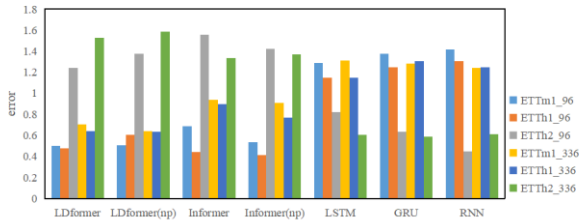


Fig. 7: Average error of the true and predicted values of the long series

In the short series prediction of Fig. 6, the LDformer has the smallest average error between the predicted and true values, and the prediction accuracy is higher than that of the other models. In Fig. 7, for long time series prediction, LDformer still achieves good results. Nevertheless, the results are unsatisfactory due to the availability of vacant data in the ETTh2 dataset. Compared with Informer, this paper combines LSTM and uses four channels to execute in parallel in the encoder to improve the stability of the model. It also uses ProbSparse self-attention combined with UniDrop to alleviate the loss of some key connections in the sequence and improve the prediction accuracy of a long series.

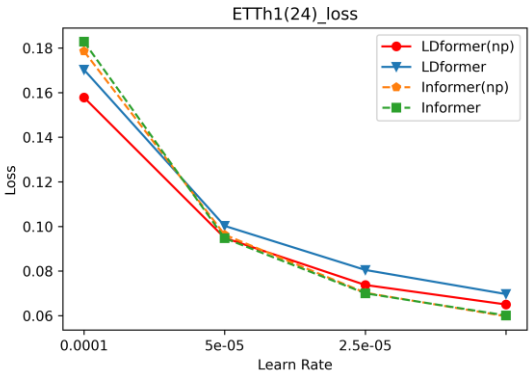


Fig. 8: Convergence of loss with decreasing learning rate

We plot the average loss variation corresponding to different learning rates in the four models. As shown in Fig. 8, the LDformer(np) proposed in this paper converges faster with the LDformer model.

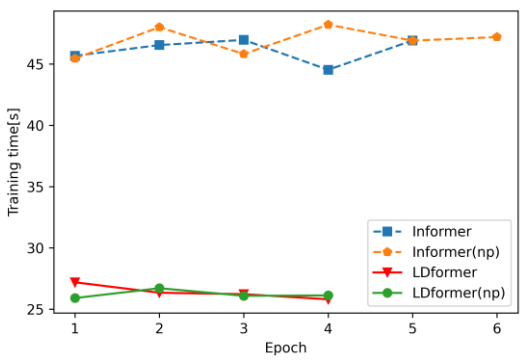


Fig. 9: Training time comparison

Fig. 9 shows the training runtime profiles of several models with increasing epochs. By the fourth epoch, this paper's model LDformer(np) and LDformer obtain the optimal results. Informer(np) obtains the optimal result at the sixth epoch. The training time of the model in this paper is significantly less than that of other models. Our proposed probabilistic sparse attention mechanism combined with UniDrop reduces the number of parameters without increasing the time complexity. Convolutional layers extract dominant features, reducing the temporal feature dimension and avoiding redundancy in the attention mechanism. This is the key factor that reduces the training time.

We performed ablation experiments for each innovation to further evaluate the effectiveness of the individual components in the LDformer. Table 6 shows the introduction table of the different modules removed, and we have named the experimental models with varying points of innovation removed separately.

We conducted experiments on four prediction tasks, comparing them with the MSE and MAE metrics. The results are shown in Table 7.

Table 7 shows that the cancellation of each innovation point affects the results. The ability of LSTM to extract deep expressions in the time series. The combination of UniDrop can alleviate the loss of key connections in the series. Multiple channels accepting long sequences in parallel can make the model structure more stable. The convolutional layer can avoid the redundancy generated by values in the attention mechanism. By combining these modules, we achieve optimal results.

Table 6: Introduction to the ablation experiment module

Model Name	LSTM	Parallel		No parallel	ProbSparse self- attention combined with UniDrop	Convolutionl layer
		2-way parallel	4-way parallel			
Model 0(M0)				✓	✓	✓
Model 1(M1)			✓		✓	
Model 2(M2)	✓			✓		✓
Model 3(M3)	✓		✓			✓
Model 4(M4)	✓			✓	✓	
Model 5(M5)	✓		✓		✓	
Model 6(M6)	✓	✓			✓	✓

Table 7: Ablation Experiment

(The best results are in bold)

Dataset	ETTm1_48		ETTm1_168		ETTh1_24		ETTh1_336	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
M0	0.1977	0.3713	0.5817	0.6859	0.4452	0.6051	0.6005	0.7049
M1	0.1719	0.3471	0.7255	0.7808	0.3028	0.4783	0.6320	0.7170
M2	0.1224	0.2849	0.5064	0.6318	0.3211	0.5132	0.9754	0.9306
M3	0.1389	0.3119	0.6188	0.7034	0.2690	0.4529	0.5424	0.6655
M4	0.0838	0.2304	0.5101	0.6338	0.2843	0.4617	0.8499	0.8273
M5	0.1400	0.3050	0.5310	0.6574	0.2574	0.4291	0.6655	0.7562
M6	0.1254	0.2916	0.6792	0.7595	0.3057	0.4964	0.6221	0.7159
LDformer(np)	0.0999	0.2574	0.5146	0.6573	0.4193	0.5937	0.5057	**0.6421**
LDformer	**0.0665**	**0.1993**	**0.5096**	**0.6166**	**0.2492**	**0.4361**	**0.5034**	0.6455

6 Conclusion

In this paper, we propose a long-term power forecasting model LDformer and its nonparallel state LDformer (np) to solve the long time series prediction problem with the power dataset ETT as an example. The LDformer model is useful for obtaining more accurate prediction results without guaranteeing an increase in time complexity. First, ProbSparse self-attention combined with UniDrop is proposed to replace sparse probabilistic attention, which can effectively avoid the risk of losing key connections between sequences without increasing the complexity. Second, two perspectives, parallel and nonparallel, are used in the encoder module for comparison. Considering the number of parameters and model stability, we use convolutional layers for data extraction between attention modules. Ultimately, this paper combines the improved model with LSTM, which is used to capture the long-range time-series information and extract the deep expressiveness in the time series to improve prediction accuracy. Experiments prove that LDformer can effectively enhance prediction accuracy. It is more suitable for long time series prediction tasks with short-interval datasets. For each innovation proposed in this paper, ablation experiments were conducted to demonstrate their feasibility. However, there is still room for improvement in the method of this paper. It is experimentally verified that the model in this paper is more suitable for long-time series prediction of short-interval datasets.

Contributors

Ran Tian: Conceptualization, Supervision, Project administration, Writing – review & editing. Xinmei LI: Methodology, Data curation, Software, Writing – original draft. Zhongyu Ma: Investigation, Writing – review & editing.Yanxing Liu: Supervision, Project administration. Jingxia Wang: Visualization, Investigation, Validation. Chu Wang: Investigation, Validation.

Compliance with ethics guidelines

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Ciechulski, T. , & Osowski, S. , 2021. High precision lstm model for short-time load forecasting in power systems. Energies, 14(11):2983-2997. http://dx.doi.org/10.3390/en14112983

Han, Z. , Zhao, J. , Leung, H. , Ma, K. F. , & Wang, W. , 2019. A review of deep learning models for time series prediction. IEEE Sensors Journal, 21(6):7833-7848. http://dx.doi.org/10.1109/jsen.2019.2923982

Zhou, H. , Zhang, S. , Peng, J. , Zhang, S. , & Zhang, W. , 2020. Informer: beyond efficient transformer for long sequence time-series forecasting. In The Thirty-Fifth AAAI Conference on Artificial Intelligence, p.1-15. http://dx.doi.org/10.48550/arXiv.2012.07436

Chuku, C. , Simpasa, A. , & Oduor, J. , 2019. Intelligent forecasting of economic growth for developing economies. International Economics, 159(1):74-93. http://dx.doi.org/10.1016/j.inteco.2019.06.001

Di, Z. , Ling, J. , Wei, Z. , Tang, K. , & Cheng, J. , 2018. Long-term traffic speed prediction based on multiscale spatio-temporal feature learning network. IEEE Transactions on Intelligent Transportation Systems, 20(10):3700-3709. http://dx.doi.org/10.1109/TITS.2018.2878068

Chakraborty, T. , Chattopadhyay, S. , & Ghosh, I. , 2019. Forecasting dengue epidemics using a hybrid methodology. Physica A: Statistical Mechanics and its Applications, 527(1):121266-121274. http://dx.doi.org/10.1016/j.physa.2019.121266

Xie, Y., & Lou, Y. , 2019. Hydrological time series prediction by ARIMA-SVR combined model based on wavelet transform. Proceddings of the 2019 3rd International Conference on Innovation in Artificial Intelligence-ICIAI 2019, p.243-247. http://dx.doi.org/10.1145/3319921.3319959

Marcjasz, G. , Uniejewski, B. , & Weron, R. , 2020. Probabilistic electricity price forecasting with narx networks: combine point or probabilistic forecasts? International Journal of Forecasting, 36(2):466-479. http://dx.doi.org/10.1016/j.ijforecast.2019.07.002

Chen, J. F. , Wang, W. M. , & Huang, C. M. , 1995. Analysis of an adaptive time-series autoregressive moving-average (arima) model for short-term load forecasting. Electric Power Systems Research, 34(3):187-196. http://dx.doi.org/10.1016/0738-7796(95)00977-1

Viccione, G. , Guarnaccia, C. , Mancini, S. , & Quartieri, J. , 2019. On the use of ARIMA models for short-term water tank levels forecasting. Water Supply, 20(3):1-13. http://dx.doi.org/10.2166/ws.2019.190

Xu, D. , Wang, Y. , Jia, L. , Qin, Y. , & Dong, H. , 2017. Real-time road traffic state prediction based on ARIMA and Kalman filter. Frontiers of Information Technology & Electronic Engineering, 18(2):287-302. http://dx.doi.org/10.1631/fitee.1500381

Khan, M. , Muhammad, N. S. , & El-Shafie, A, 2020. Wavelet based hybrid ann-arima models for meteorological drought forecasting. Journal of Hydrology, 590(1):125380-125389. http://dx.doi.org/10.1016/j.jhydrol.2020.125380

Syriopoulos, T. , Tsatsaronis, M. , & Karamanos, I, 2020. Support vector machine algorithms: an application to ship

price forecasting. Computational Economics, 57(1):1-33. http://dx.doi.org/10.1007/s10614-020-10032-2

Min, D. , Hao, Z. , Hua, X. , Min, W. , Kzl, C. , & Yn, D. , et al, 2020. A time series model based on hybrid-kernel least-squares support vector machine for short-term wind power forecasting. ISA Transactions, 108(1):1-11. http://dx.doi.org/10.1016/j.isatra.2020.09.002

Nobrega, J. P. , & Oliveira, A, 2019. A sequential learning method with kalman filter and extreme learning machine for regression and time series forecasting. Neurocomputing, 337(14):235-250. http://dx.doi.org/10.1016/j.neucom.2019.01.070

Chen, X., & Sun, L, 2021. Bayesian temporal factorization for multidimensional time series prediction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(9):1-15. http://dx.doi.org/10.1109/tpami.2021.3066551

Kim, S. H. , Lee, G., Kwon, G. Y., Kim, D. I., & Shin, Y. J, 2018. Deep learning based on multi-decomposition for short-term load forecasting. Energies, 11(12):3433-3450. http://dx.doi.org/10.3309/en11123433

Guo, S. , Lin, Y. , Li, S. , Chen, Z. , & Wan, H, 2019. Deep spatial-temporal 3d convolutional neural networks for traffic data forecasting. IEEE Transactions on Intelligent Transportation Systems, 20(10):3913-3926. http://dx.doi.org/10.1109/tits.2019.2906365

Hosseini, M. K. , & Talebpour, A, 2019. Traffic prediction using time-space diagram: a convolutional neural network approach. Transportation Research Record: Journal of the Transportation Research Board, 2673(7):425-435. http://dx.doi.org/10.1177/0361198119841291

Jiasheng, C. , & Jinghan, W, 2019. Stock price forecasting model based on modified convolution neural network and financial time series analysis. International Journal of Communication Systems, 32(12):1-13. http://dx.doi.org/10.1002/dac.3987

Hu, R. , Chiu, Y. C. , & Hsieh, C. W, 2020. Crowdedness prediction on mass rapid transit systems using a weighted bidirectional recurrent neural network. IET Intelligent Transport Systems, 14(3):196-203. http://dx.doi.org/10.1049/iet-its.2018.5542

Min, K. , Kim, D. , Park, J. , & Huh, K, 2019. Rnn-based path prediction of obstacle vehicles with deep ensemble. IEEE Transactions on Vehicular Technology, 68(10):10252-10256. http://dx.doi.org/10.1109/tvt.2019.2933232

Zhang, L. , Wang, G. , &Giannakis, G. B, 2019. Real-Time power system state estimation and forecasting via deep unrolled neural networks. IEEE Transactions on Signal Processing, 67(15):4067-4077. http://dx.doi.org/10.1109/tsp.2019.2926023

Xiao, S. , Yan, J. , Farajtabar, M. , & Song, et al, 2019. Learning time series associated event sequences with recurrent point process networks. IEEE transactions on neural networks and learning systems, 30(10):3124-3136. http://dx.doi.org/10.1109/tnnls.2018.2889776

Zheng, J. , & Huang, M. , 2020. Traffic flow forecast through time series analysis based on deep learning. IEEE Access, 8(1):83562-82570. http://dx.doi.org/10.1109/access.2020.2990738

Karevan, Z. , & Suykens, J. A. K. ,2020. Transductive LSTM for time-series prediction: An application to weather forecasting. Neural Networks. 125(1):1-9. http://dx.doi.org/10.1016/j.neunet.2019.12.030

Zhang, T. , Song, S. , Li, S. , Ma, L. , Pan, S. , & Han, L. , 2019. Research on gas concentration prediction models based on LSTM multidimensional time series. Energies. 12(1):161-176. http://dx.doi.org/10.3390/en12010161

Kcm, B. , Tth, A. , Yqy, B. , Hui, L. A. , Peng, C. , & Bing, W. D. , et al, 2020. Application of lstm for short term fog forecasting based on meteorological elements. Neurocomputing, 408(30):285-291. http://dx.doi.org/10.1016/j.neucom.2019.12.129

Gai, S. , Zeng, X. , & Yuan, T. , 2021. Parking volume forecast of railway station garages based on passenger behaviour analysis using the lstm network. Journal of Advanced Transportation, 2021(6688609):1-14. http://dx.doi.org/10.1155/2021/6688609

Ran, X. , Shan, Z. , Fang, Y. , & Lin, C. , 2019. An lstm-based method with attention mechanism for travel time prediction. Sensors, 19(4):861-883. http://dx.doi.org/10.3390/s19040861

Wang, Z. , Qu, J. , Fang, X. , Li, H. , & Ren, H. , 2020. Prediction of early stabilization time of electrolytic capacitor based on ARIMA-Bi_LSTM hybrid model. Neurocomputing, 403(1):63-79. http://dx.doi.org/10.1016/j.neucom.2020.03.054

Khandelwal, U. , He, H. , Qi, P. , & Jurafsky, D. , 2018. Sharp nearby, fuzzy far away: how neural language models use context. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics,p.1-14. http://dx.doi.org/10.48550/arXiv.1805.04623

Wu, N. , Green, B. , Xue, B. , & O'Banion, S. , 2020. Deep transformer models for time series forecasting: the influenza prevalence case. the 37th International Conference on Machine Learning, Vienna, Austria, p.1-10. http://dx.doi.org/10.48550/arXiv.2001.08317

Li, S. , Jin, X. , Xuan, Y. , Zhou, X. , Chen, W. , & Wang, Y. X. , et al. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. NeurIPS, p.1-14. http://dx.doi.org/10.48550/arXiv.1907.00235

Vaswani,A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., &Polosukhin,I. 2017. Attention is all you need. 31st Conference on Neural Information Processing Systems(NIPS 2017). p.5998-6008. http://dx.doi.org/10.48550/arXiv.1706.03762

Wu, Z. , Wu, L. , Meng, Q. , Xia, Y. , Xie, S. , & Qin, T. , et al. 2021. Unidrop: a simple yet effective technique to improve transformer without extra cost. NAACL, p.1- 14. http://dx.doi.org/10.48550/arXiv.2104.04946

Wang, C. , Tian, R. , Hu, J. , Ma, Z. ,2023. A Trend Graph Attention Network for Traffic Prediction. Information Sciences, 623(1):275-292. http://dx.doi.org/10.1016/j.ins.2022.12.048