Improving Accuracy and Efficiency in Time Series Forecasting with an Optimized Transformer Model

Junhong Chen, Hong Dai*, Shuang Wang, and Chengrui Liu

***Abstract*—Time series forecasting (TSF) is a prevalent research task in various fields such as medicine, transportation, environment, network detection, finance, and others. The TSF task aims to identify underlying patterns in data and make relatively accurate estimates of future data based on known values. In recent years, deep learning models have gained popularity for TSF tasks due to their capability to capture internal information effectively. However, traditional deep-learning models encounter difficulties when parallelizing data calculations, leading to error accumulation and reduced forecasting accuracy. Additionally, when dealing with excessively long input data, traditional deep learning models may experience performance degradation despite providing sufficient information and making it arduous to predict future data. Transformer-based models, with Self-Attention as the core, have shown the ability to facilitate global information interaction and enhance prediction efficiency. Nonetheless, they may encounter problems with significant and redundant parameters, causing unnecessary time overhead. To overcome these challenges, we propose a novel model called VarSeg-Trans, which incorporates three key optimizations: the cut-up mechanism, the variables-isolating mechanism, and an improved attention calculation method to enhance the transformer model's performance. Specifically, the cut-up mechanism enables the model to process longer input sequences, the variables-isolating mechanism mitigates overfitting, and the improved attention method leverages sequence information more effectively. Compared to other baseline TSF models and previous Transformer-based models, VarSeg-Trans has achieved an average reduction of 9% in MSE and MAE, along with a 3% increase in the coefficient of determination R^2. This trend is substantiated by consistent results across multiple experimental trials.**

***Index Terms*—Time series forecasting, Deep learning, Transformer-based models, Self-Attention**

I. Introduction

TIME series forecasting (TSF), particularly long-term TSF, is essential in various real-life scenarios. Such as predicting environment quality using a hybrid Convolutional Neural Network - Long Short Term Memory (CNN-LSTM) model [1–3], estimating traffic flow using Graph Convolution Recurrent Network (GCRN) [4–6], forecasting stock prices using an ARIMA-based model [7–9], forecasting electricity demand using Recurrent Neural Networks (RNNs) [10–12], and predicting air quality and population movement using Support Vector Machines(SVM) [13, 14]. Unfortunately, most existing models are designed for short-term forecasting, typically predicting points in less than 24 steps.

Manuscript received May 11, 2023; revised Oct 20, 2023. The research work was supported by Graduate student science and technology innovation project of University of Science and Technology Liaoning (No. LKDYC202221).

Junhong Chen is a Postgraduate of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, China. (e-mail: shopkeeperakashi@hotmail.com).

Hong Dai* is a Professor of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, China. (corresponding author to provide phone: +086-186-4226-8599; fax: 0412-5929818; e-mail: dear_red9@163.com).

Shuang Wang is a Postgraduate of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, China. (e-mail: 1657669526@qq.com).

Chengrui Liu is a Postgraduate of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, China. (e-mail: cherryliumark@163.com).

TABLE I
LSTM Performance and Forecasting Length

Different forecasting steps	MSE	MAE
12	0.173	0.167
24	0.272	0.263
36	0.643	0.632
48	1.505	1.528
96	3.831	4.002

Take the LSTM, a wide-used model developed from RNNs, as an example in Table I. As the sequence length increases, the influence of historical data on the forecast output may weaken, thereby increasing the forecast error. When forecasting more than 24 steps, the MSE of the result can increase significantly [15]. Therefore, researchers need to explore new models and techniques to improve the accuracy of long-term TSF.

To summarize, here are some of the difficulties associated with TSF tasks:

1) Long-term dependencies: In time series prediction, the current data is often related to multiple past time points, and these relationships may be long-term. Deep learning models usually have strong representation capabilities and can automatically learn long-term dependencies.
2) Non-stationarity: Time series data is usually non-stationary, meaning its statistical characteristics change over time. This characteristic can make the prediction task more challenging.
3) Multivariate prediction: In most cases, time series prediction requires considering not only past historical data but also the influence of other variables that may affect the data, making the model more complex.

In order to achieve better prediction results, we conducted numerous experiments to validate our findings. The contributions of this paper are as follows:

1) We introduced a cut-up mechanism to address the problem of insufficient information per time step. Implementing this mechanism enhances the module's efficiency in processing longer sequence inputs and uncovering more relationships.
2) We introduced a variables-isolating mechanism that involves predicting each variable individually. The prediction results are then stitched together in the end. This approach allows the Transformer model to focus on a single variable, resulting in faster calculations.
3) We improved the core calculation mechanism of the existing excellent model to enhance both forecasting time and prediction accuracy.

II. Related work

A. Transformer Modules and Their Variants

Since the Vanilla Transformer model was introduced in 2017 [16], the structure diagram is shown in Fig. 1. Many fields, including Natural Language Processing (NLP) and Computer Vision (CV), have made breakthroughs using the Transformer model or its core method. Numerous Transformer-based models have also been introduced in TFS tasks. The Self-Attention mechanism of the Transformer model, which enables it to efficiently extract features from time series and capture dependencies among elements in long sequences [17, 18], has significantly improved the accuracy of long-term forecasting. Reformer [19], Informer [20], and Autoformer [21] are excellent variants of the Transformer model that demonstrate outstanding performance in TFS tasks.

However, all Transformer-based models face some common problems [22, 23]. Firstly, the Self-Attention mechanism is a time-consuming process. In Fig. 2, we depict the computation of the Self-Attention mechanism, where it becomes evident that 16 computations are required to consider merely 4 input elements. As a consequence of the imperative requirement to perform pairwise comparisons of attention weights between each position and all other positions within the input sequence during the computation process, a quadratic computational complexity arises, leading to a pronounced escalation in both computational and memory overhead as the length of the input sequence undergoes substantial extension. In other words, the temporal and spatial complexities manifest as $O(L^2)$. Even a relatively short input will bring huge costs. Secondly, if the model reads too much input sequence, it can fall into a local optimal solution. Based on the previous, this can increase computational cost. However, if too little data is read, it will fail to build a model. Thirdly, Unlike NLP tasks, TSF tasks exhibit a notable distinction in which the informational content encapsulated within a solitary time step is circumscribed, thereby limiting the capacity of individual time steps to convey substantial significance. These three aspects impede the applicability of Transformer-based models in the context of TFS tasks.

To address these challenges, researchers have proposed various methods. Reformer addresses these issues using

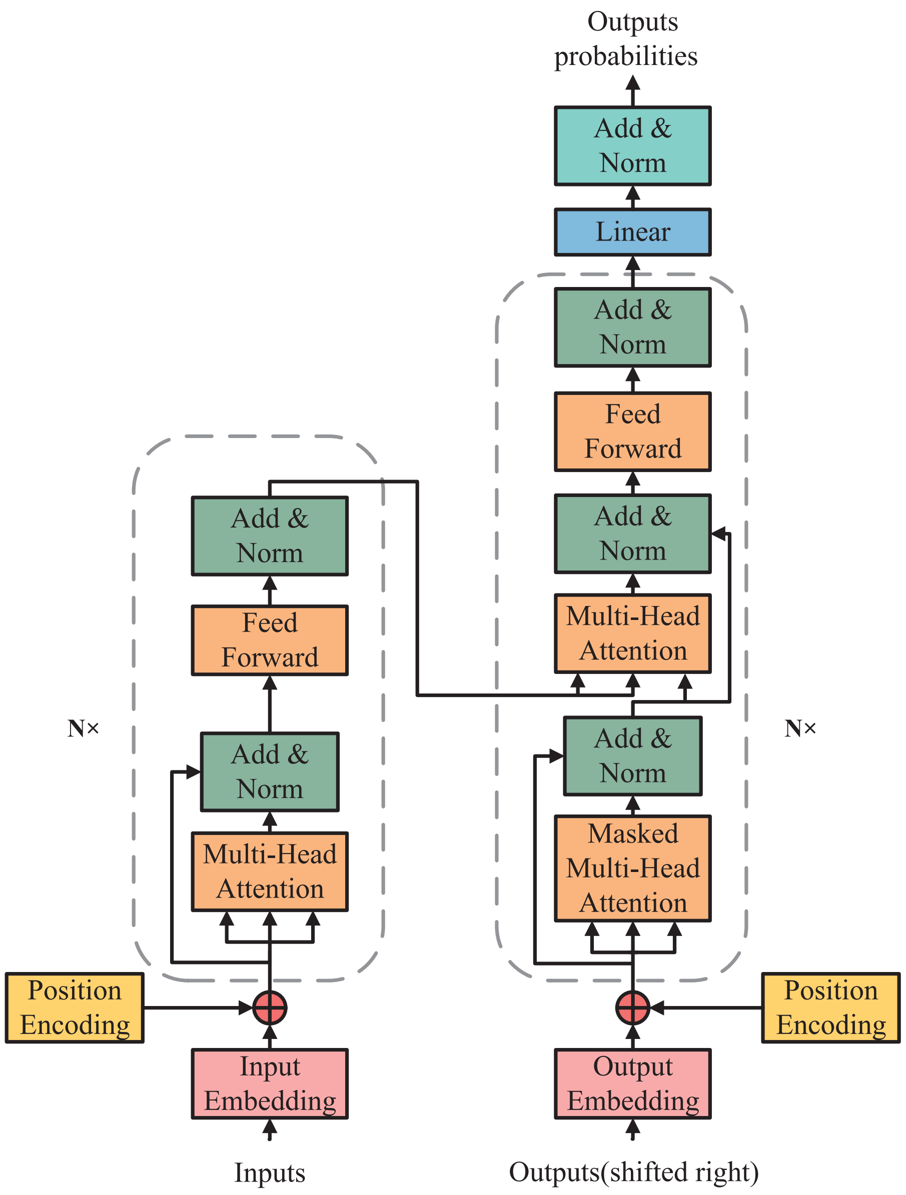

Fig. 1. Overview of Transformer Model Structure

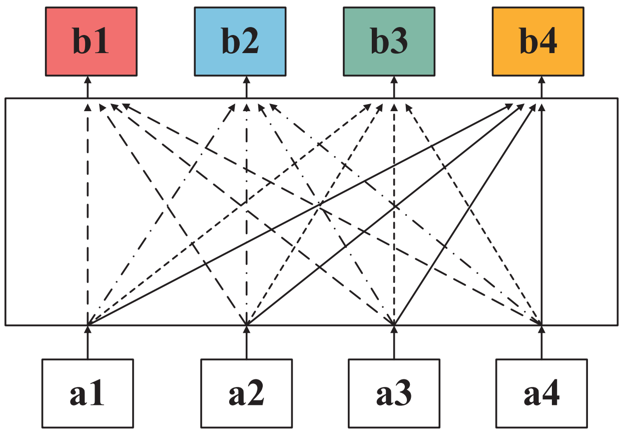

Fig. 2. Illustration of Self-Attention Computation Complexity

Locality Sensitive Hashing Attention and Chunking Feed-forward Neural Network (HA-CFNN) layers to reduce time and space costs. However, this approach only works when the input sequence is excessively long. Informer proposes a ProbSparse Self-Attention calculation and Self-Attention Distilling, which can reduce computational cost. However, it may limit the efficiency of information utilization and affect the prediction quality. Autoformer overcomes the bottleneck of information utilization through the Decomposition Architecture and Auto-Correlation mechanism. However, it spends too much time searching for the periodic characteristics of time series data, making it unsuitable for training on datasets with weak periodicity.

Therefore, to overcome the abovementioned problems, we have designed a new model that can better handle TFS tasks. We have made several improvements to the Transformer-based TFS model and proposed several structural changes to increase forecasting accuracy.

B. *Variables-isolating Mechanism in Other Fields*

Variables-isolating, which involves processing data in different dimensions, is used in various fields. One such field is CV, where AlexNet performs independent convolution operations on different channels to improve the model's accuracy and efficiency [24]. Moreover, Ioffe and Szegedy introduced a batch normalization operation, which performs independent normalization on each channel to reduce internal covariate shifts and enhance the robustness of the model [25]. However, despite the widespread use of this technique in CV, it has been scarcely adopted in TSF tasks.

C. *Cut-up Mechanism in Other Fields*

Cut-up involves splitting a sequence into smaller fragments and has been widely used in various deep-learning models. In NLP, Taku Kudo was the first to adopt subword tokenization with an unigram language model [26]. While BERT, the most famous example of this technique, uses subword-based tokenization instead of traditional character-based tokenization [27]. In CV, YOLO divides the input image into multiple small blocks for processing, which improves the speed and accuracy of the model's detection capabilities [28, 29]. Similarly, UTNet has made significant progress in the medical image field by dividing the input image into multiple small blocks for processing [30–32]. This approach can also be applied to TFS tasks by splitting the input sequence into smaller parts, where each part represents one token instead of a one-time step representing one token.

III. Preliminary

To begin with, we should clearly define TSF tasks. Given a set of observation values $X_{d\times L} = [x_{t_1}, x_{t_2}, \ldots, x_{t_L}]$ from time t_1 to t_L and the corresponding real values $Y_{d\times T} = \left[y_{t_{L+1}}, y_{t_{L+2}}, \ldots, y_{t_{L+T}}\right]$ from time t_{L+1} to t_{L+T}. Our objective is to locate a function or model that satisfies Equation (1):

$$f(x_{t_1},\ x_{t_2}, \ldots\ x_{t_L}) = [\hat{y}_{t_{L+1}},\ \hat{y}_{t_{L+2}},\ \ldots, \hat{y}_{t_{L+T}}] \qquad (1)$$

Naturally, all previous work and our work should aim to minimize the gap between the real values $\left[y_{t_{L+1}}, y_{t_{L+2}}, \ldots, y_{t_{L+T}}\right]$ and forecasting values $[\hat{y}_{t_{L+1}},\ \hat{y}_{t_{L+2}}, \ldots, \hat{y}_{t_{L+T}}\ ]$. The L and T represent the length of the observed and forecasted values, which may not be equal in typical circumstances. The d in observation value $X_{d\times L}$ and real value $Y_{d\times T}$ represents dimension that may also differ. The d in the real values is typically 1. While in the observed values, it is usually greater than 2.

One of the most well-known models for TSF tasks is the Transformer, which is based on an exciting mechanism called Self-Attention. Like humans, the Transformer model prioritizes specific parts of data using the Self-Attention mechanism. This mechanism can accept n inputs that impact each other and then identify the significant points for attention score calculation. The output from the module is the attention score of these interactions.

The Transformer model relies heavily on the attention mechanism to capture global dependencies between input and output. As depicted in Fig. 1, the Self-Attention module is the cornerstone of both the encoder and decoder modules. Notably, the encoder and decoder are similar, with minor differences in structure. Unlike the RNN model, which encounters the vanishing gradient problem due to diminishing gradients during training and requires $n-1$ steps to process the $n-th$ input, the Transformer and Transformer-based models have a direct path of only 1, thereby circumventing the vanishing gradient issue. The Transformer model's remarkable capacity to capture long-term dependencies and interactions confers substantial advantages for TSF tasks.

IV. Methods

We proposed a new model to solve the mentioned problems, which includes a variables-isolating mechanism and cut-up mechanism, as well as an improved Transformer model. The specific flowchart can be seen in Fig. 3.

A. Variables-isolating Mechanism

The multivariate time series represents a dataset with multiple variables. Each time step may contain a diverse set of information. In traditional Transformer-based models, all variables are read simultaneously. This can make input tokens with multiple dimensions and increase computational complexity.

Correspondingly, a variables-isolating mechanism is employed to partition the input value $X_{d\times L}$ split into $(X_1, X_2, \ldots, X_d)$. Each X_i (where $i \in 1,\ 2,\ \ldots,\ d$) denotes a single feature in $X_{d\times L}$ with length L. Consequently, a one-dimensional input sequence is formed, which the Transformer model processes. Each Transformer module can handle the input more efficiently and lower computational costs by processing univariate sequences with shared weights across all dimensions.

We proposed a learnable weighting vector denoted as $W_{1\times d}$, possessing a dimension equivalent to the number of variables within the inputs. Each constituent element of this vector signifies the weight assigned to its corresponding variable, serving the purpose of weighting and amalgamating the prediction results associated with each variable. Independent predictions for each variable undergo multiplication by this designated weighted vector, with the resultant values subjected to summation, thus engendering an overarching prediction. This process can be expressed as Equation (2):

$$R_{Vi} = \sum_{i\ =\ 1}^{d} w_i \times \hat{y}_i \tag{2}$$

Within this equation, R_{Vi} means the final weighted average summation result, and w_i denotes the value at the i_{th} position of the weight vector, while $\hat{y}_i$ represents the predicted value of the i_{th} variable. Initially set to a uniform value of 1, the weights are regarded as learnable parameters, amenable to optimization throughout model training to minimize prediction errors or other pertinent metrics.

Incorporating a variables-isolating mechanism in TSF tasks can result in three major benefits [33]. Firstly, it enables a simplified model that predicts each variable independently without influences from other variables. Secondly, the overall accuracy of time series prediction can be improved by using different prediction methods for each variable, given that the prediction of each variable is independent. Thirdly, feature engineering can be simplified by applying different feature engineering methods to each variable independently without considering the impact of other variables. Thereby can reduce the complexity of feature engineering.

B. Cut-up Mechanism

Through the application of variables-isolating, standardization, and normalization operations, the module divides the input into fixed-length segments. It discards any remaining data that does not meet the minimum length requirement. However, determining the appropriate length involves various factors [34], such as data length, the information density of the data itself, time scale, and task requirements. Fig. 4(a) indicates that the modeling time decreases as the segment length increases. Fig. 4(b) demonstrates that the accuracy is the highest when the segment length is 32. The error bar in Fig. 4(a) is the standard deviation of five experiments.

Assuming the initial input possesses a length denoted as L equivalent to the extent of observation values, the number of segments following this partitioning is designated as N, with each segment uniformly possessing a length denoted as l. In order to optimize the exploitation of information within the time series data, an overlapping arrangement is instituted among these segments, characterized by the parameter o. Then, we can get a simple equation: $N = L/(l-o)$. This equation means the cut-up mechanism can reduce the original input sequence's length to $1/(l-o)$. When o is set to 0, the input length can be significantly reduced to $1/l$, enabling the model to process more input data while maintaining the same computational complexity.

Each segment is treated as a single token. The model will hand them over to the improved Transformer model for further processing to obtain the prediction result. There are two reasons for using the cut-up mechanism. Firstly, time series exhibit strong locality, meaning adjacent values are highly similar. Therefore, it is reasonable to base the attention calculation on segments. Secondly, the cut-up mechanism can improve forecasting performance by allowing the model to read more extended observable sequences.

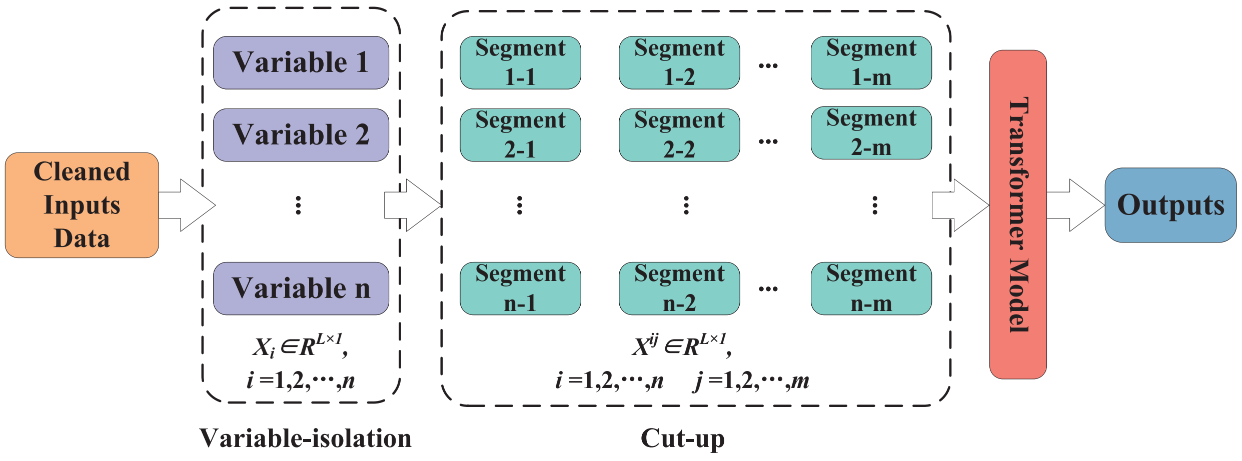

Fig. 3. Overview of the Proposed Model's Workflow

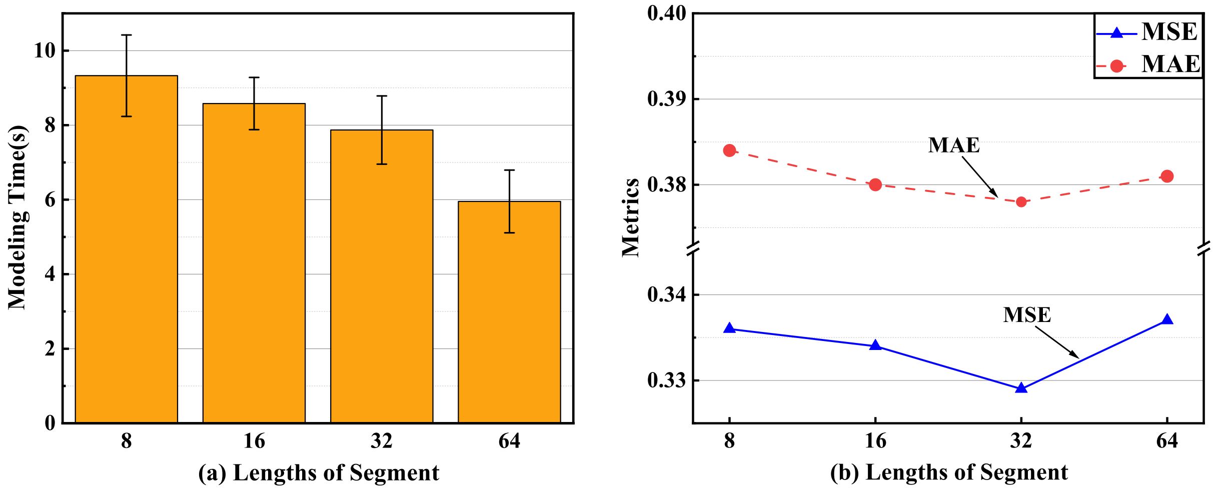

Fig. 4. Variation in Evaluation Metrics and Modeling Time Across Different Segment Lengths

C. Transformer Model

We employ the Transformer's encoder module to extract time dependencies in time series. First, applying the linear transformation matrix $W_L \in R^{L\times N}$ and the position encoding matrix $W_{PE} \in R^{L\times D}$ shown in Equation (3).

$$PE_{(pos,\ 2i)} = \sin(\frac{pos}{1000^{\frac{2i}{d_k}}}) \\ PE_{(pos,\ 2i+1)} = \cos(\frac{pos}{1000^{\frac{2i}{d_k}}}) \tag{3}$$

Then, the model utilizes the learnable weight matrices $W_Q \in R^{D\times d_k}$, $W_K \in R^{L\times d_k}$, $W_V \in R^{L\times d_k}$ to convert input X_i into $Query(Q)$, $Key(K)$, $Value(V)$, where d_k represents the embedding dimension. The conversion process is shown in Equation (4).

$$Q_j = W_Q^j X_i^T,\ j=1,\ 2,\ \dots,\ 8 \\ K_j = W_K^j X_i^T,\ j=1,\ 2,\ \dots,\ 8 \\ V_j = W_V^j X_i^T,\ j=1,\ 2,\ \dots,\ 8 \tag{4}$$

Attention scores are obtained after performing a scaled dot-product operation of Equation (5) among Q, K, and V. In this equation, $Q \in R^{L_Q\times d}$, $K \in R^{L_K\times d}$, $V \in R^{L_v\times d}$, d means dimension.

$$Attention\,(Q,\ K,\ V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

Further improvements will be discussed in the following section. In addition, the multi-head attention mechanism is employed to enhance the model's performance. As shown in Equation (6), the Transformer model will calculate multiple sets of Q, K, and V.

$$Q_i = W_Q\ (W_L X_i^T + W_{PE} X_i^T\ ), \quad Q_i \in R^{L\times d_k} \\ K_i = W_K\ (W_L X_i^T + W_{PE} X_i^T\ ), \quad K_i \in R^{L\times d_k} \\ V_i = W_V\ (W_L X_i^T + W_{PE} X_i^T\ ), \quad V_i \in R^{L\times d_k} \tag{6}$$

This mechanism extends the Self-Attention mechanism by projecting input feature vectors onto multiple attention heads, with the vanilla Transformer defaulting to 8. That is as shown in Equation (7), calculate 8 groups of attention mechanisms.

$$head_j = Attention\,(Q_j,\ K_j,\ V_j)\,,\ j=1,\ 2,\ \dots,\ 8 \tag{7}$$

Attention weights and outputs are calculated for each head, and the outputs from all attention heads are concatenated to form the final output. This process is as described in Equation (8). Different Q, K, and V increase the model's expressive power, and computation can be parallelized. Furthermore, multi-head attention can enable the model to learn distinct feature representations in different subspaces, each focusing on a specific aspect of attention.

$$MutilHeadAttention\,(Q,\ K, V) \\ = Concat(head_1,\ head_2, \dots,\ head_8\ ) \tag{8}$$

After the multi-head attention calculation, the data is propagated through a batchNorm layer and a feed-forward network layer with residual connections. The final output is obtained via a feed-forward layer, which computes a feed-forward calculation on the Self-Attention layer's output. The resulting output is denoted by $\hat{Y} = (\hat{y}_{t_{L+1}},\ \hat{y}_{t_{L+2}},\ \dots, \hat{y}_{t_{L+T}})$.

D. Dynamic ProbSparse Attention Calculation Mechanism

The Informer proposed the ProbSparse Attention mechanism to enhance the efficiency and storage of Self-Attention computations. This mechanism only considers interactions between a small subset of adjacent positions in the input sequence rather than comparing all positions. The adjacent positions are selected based on dot-product similarity calculations where only the most relevant positions are included in the interaction. Consequently, the attention weight matrix generated from this approach contains a sparse subset of non-zero elements, significantly reducing computational and storage costs. Informer uses the Equation (9) to measure the $i-th\ Q's$ sparsity.

$$M\,(Q_i\ K) = ln\sum_{j=1}^{L_K} e^{\frac{Q_i K_j^T}{\sqrt{d}}} - \frac{1}{L_k}\sum_{j=1}^{L_K}\frac{Q_i K_j^T}{\sqrt{d}} \tag{9}$$

Based on this measurement, the ProbSparse Attention converts the Self-Attention Equation (5) into the following Equation (10). $\bar{Q}$ is a sparse matrix with the same size as Q and it contains only queries that are active in Q_L,where L means the length of Q.

$$Attention\left(Q,\ K,\ V\right) = softmax\left(\frac{\bar{Q}K^T}{\sqrt{d_k}}\right)V \qquad (10)$$

However, the ProbSparse Attention mechanism possesses several drawbacks:

1) Being a sparse attention mechanism, it may overlook critical inputs, leading to inaccurate results and inefficient use of information.
2) Its inability to handle long sequences is widespread among Transformer-based models.
3) Sparse operations can have adverse effects on sparse input sequences.

As a remedy, we propose a dynamic ProbSparse Attention calculation mechanism. This mechanism switches to the full-attention calculation if it detects a Q that appears too dense. Conversely, if the $Q's$ sparsity conforms to Equation (2), the mechanism will use the ProbSparse attention calculation. Moreover, we will use half of L_Q instead of lnL_Q to enhance information utilization.

V. Experiment

A. Datasets

VarSeg-Trans has been assessed on multiple popular datasets, including Weather, Electricity, and ETT (ETTh1, ETTh2, ETTm1). These datasets typically exhibit higher timesteps and more excellent stability, decreasing susceptibility to overfitting. Table II summarizes the statistical information about these datasets.

TABLE II
Overview of Experiment Datasets

Datasets	Number of Features	Number of Instances	Start Date -End Date	Duration (Months)
ETTh1	8	17520	2016.07-2018.07	24
ETTh2	8	8760	2016.07-2018.07	24
ETTm1	8	70080	2016.07-2018.07	24
Weather	13	35065	2010.01-2013.12	47
Electricity	9	2075259	2006.12-2010.11	47

1) ETT: This dataset contains historical temperature and weather data for ten power transformers in a particular region of China. Each transformer has multiple sensors that record temperature values at different positions. Additionally, the dataset includes daily weather conditions such as temperature, humidity, wind speed, and weather types. The primary use of the ETT dataset is for predicting the temperature of power transformers, aiding engineers and researchers in forecasting future temperature trends.The ETT dataset has three different time scales, the 2-hour level, 1-hour level, and 15-minute level, which we refer to as ETTh2, ETTh1, and ETTm1, respectively. The dataset is split into train/val/test sets of 12/4/4 months.
2) Weather: This dataset collects hourly climate data from 1,600 locations in the United States from 2010 to 2013, including the forecast target "wet bulb" and 11 other climate indicators. The train/val/test is 28/10/10 months.
3) Electricity: This dataset comprises 2,075,259 measurements of residential electricity consumption in kilowatt-hours, recorded at 15-minute intervals, within a household located in Sceaux, a vicinity situated 7 kilometers from Paris, France, spanning from December 2006 to November 2010.

B. Baselines and Experimental Setting

To assess the performance of VarSeg-Trans in the context of time series forecasting (TSF), we have chosen to compare it against three Transformer-based TSF models and the LSTM model, a widely employed baseline in the TSF domain. A concise overview of these baseline models is provided below:

1) Autoformer: In essence, Autoformer operates on a principle akin to the Transformer architecture. Its distinguishing feature lies in its fusion of the advantages of the self-attention mechanism, a core component of the Transformer, and autoregressive modeling. The self-attention mechanism enables the model to consider interdependencies among different positions when processing sequential data. At the same time, autoregressive modeling empowers the model to predict the next element based on the known sequence.
2) Informer: The Informer model is a deep learning architecture designed explicitly for time series forecasting. Its primary advantage lies in effectively capturing and predicting temporal patterns within sequential data types. Key features of the Informer model include its efficient Transformer architecture, enabling the handling of long-term dependencies, and adaptability to irregularly sampled time series data. The architecture of Informer incorporates components such as the ProbSparse Attention mechanism, temporal convolution layers, and autoregressive forecasting. Collectively, these attributes provide robust tools and methodologies for its application in time series analysis.
3) Reformer: Reformer, a deep learning model based on the Transformer architecture, is characterized by its remarkable capacity to preserve high performance while substantially mitigating the demands on memory and computational resources. In the context of large-scale sequences, Reformer introduces the concept of local hashed attention, a technique that effectively reduces computational complexity. Furthermore, Reformer incorporates sparse attention mechanisms, significantly lowering self-attention's computational cost, rendering it particularly suitable for extended sequences. Additionally, Reformer can enhance its model's performance through a trainable sorting mechanism applied to input sequences.
4) LSTM: LSTM is a deep learning model widely employed in time series forecasting tasks. Its principal characteristic lies in its internal architecture, which facilitates the effective capture and utilization of long-term dependencies within time series data. The core components of an LSTM model include memory cells, each endowed with the ability to store and update information. This distinctive structure enables the model to retain information relevant to past data points, which

TABLE III
VARSEG-TRANS AND OTHER BASELINE MODELS COMPARISON

Datasets	Forecasting Steps	VarSeg-Trans			Autoformer			Informer			Reformer			LSTM		
		MSE	MAE	R^2	MSE	MAE	R^2	MSE	MAE	R^2	MSE	MAE	R^2	MSE	MAE	R^2
ETTh1	24	0.334	**0.380**	**0.942**	**0.329**	0.422	0.937	0.340	0.431	0.921	0.367	0.427	0.892	0.403	0.504	0.883
	48	**0.377**	0.453	**0.911**	0.393	0.483	0.887	0.407	0.450	0.891	0.392	0.472	0.875	0.620	0.715	0.827
	96	**0.418**	**0.449**	**0.874**	0.437	0.451	0.854	0.449	0.464	0.847	0.440	0.543	0.799	0.966	1.288	0.744
	192	**0.449**	**0.537**	**0.834**	0.464	0.542	0.808	0.522	0.623	0.796	0.563	1.204	0.746	1.513	1.609	0.629
ETTh2	24	**0.129**	**0.201**	**0.924**	0.182	0.237	0.918	0.159	0.236	0.919	0.194	0.231	0.885	0.378	0.594	0.853
	48	**0.147**	**0.212**	**0.908**	0.196	0.253	0.888	0.188	0.249	0.889	0.224	0.268	0.838	0.593	0.654	0.823
	96	**0.186**	**0.253**	**0.850**	0.204	0.284	0.843	0.197	0.278	0.829	0.246	0.322	0.804	0.874	1.106	0.759
	192	**0.210**	**0.265**	**0.822**	0.254	0.301	0.805	0.249	0.320	0.771	0.270	0.310	0.742	1.447	1.533	0.620
ETTm1	24	0.253	**0.277**	**0.948**	0.276	0.329	0.900	**0.250**	0.301	0.912	0.283	0.314	0.912	0.411	0.624	0.885
	48	**0.340**	**0.373**	**0.927**	0.383	0.426	0.870	0.372	0.394	0.865	0.377	0.454	0.861	0.704	0.902	0.822
	96	**0.382**	**0.402**	**0.859**	0.411	0.464	0.834	0.495	0.554	0.813	0.528	0.548	0.790	1.095	1.253	0.762
	192	**0.432**	0.513	0.801	0.458	**0.479**	**0.806**	0.501	0.625	0.770	0.542	0.634	0.742	1.471	1.563	0.615
Weather	24	0.117	0.178	**0.931**	0.119	0.173	0.881	**0.115**	**0.168**	0.927	0.204	0.239	0.890	0.267	0.602	0.859
	48	**0.162**	**0.211**	0.874	0.173	0.229	**0.884**	0.187	0.217	0.861	0.222	0.271	0.858	0.409	0.720	0.809
	96	**0.204**	**0.237**	**0.868**	0.228	0.331	0.829	0.241	0.340	0.838	0.249	0.304	0.801	0.896	1.247	0.759
	192	**0.242**	**0.293**	**0.806**	0.263	0.374	0.795	0.270	0.390	0.785	0.288	0.326	0.768	1.337	1.554	0.617
Electricity	24	0.143	0.184	0.940	0.147	0.168	0.906	0.139	**0.167**	0.939	0.183	0.211	0.878	**0.137**	0.173	**0.952**
	48	**0.168**	0.209	**0.876**	0.178	**0.202**	0.857	0.190	0.228	0.867	0.194	0.250	0.841	0.300	0.346	0.791
	96	**0.182**	**0.266**	**0.851**	0.195	0.317	0.803	0.204	0.345	0.824	0.224	0.336	0.777	0.732	0.843	0.737
	192	**0.217**	**0.293**	**0.829**	0.234	0.323	0.763	0.241	0.378	0.761	0.239	0.370	0.749	1.249	1.609	0.616

can be referenced in subsequent time steps. Additionally, LSTM leverages gating units to control the flow of information rigorously. Lastly, the multi-layered structure of LSTM permits the model to learn and represent the inherent properties of time series data at various levels of abstraction.

We use Python and Pytorch deep learning architecture to implement the above five models. The model training and validation were performed on a Linux server with an Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz and an NVIDIA RTX 2080Ti (11GB) GPU. To evaluate their performance, we will set the length of the observation values L=96 and the length of forecasting values T∈{24, 48, 96, 192}. In addition, we partitioned the five datasets into training, validation, and test sets in a ratio of 6:2:2, respectively. In the case of Transformer-based models, it is necessary to set the number of encoders and decoders to be the same and ensure that other parameters, such as the sizes of Q, K, and V, are the same. We run each model five times for each dataset and use the average value as the result.

C. Evaluation Metrics

We use three evaluation indicators, MAE, MSE, and R-squared, to evaluate the difference between a model's forecasting and actual values. The MSE is calculated by subtracting the predicted value from the actual value, squaring the difference, and taking the mean. The MAE is similar to the MSE, but instead of squaring the difference. It takes the absolute value of the difference. The calculation Equation (11) is as follows:

$$\begin{aligned} MSE &= \frac{1}{n}\sum_{i=1}^{n}(\hat{y}-y)^2 \\ MAE &= \frac{1}{n}\sum_{i=1}^{n}|\hat{y}-y| \end{aligned} \tag{11}$$

R-squared (R²), also referred to as the coefficient of determination, serves as a statistical metric for assessing a regression model's efficacy in fitting observed data. It quantifies the extent to which a regression model elucidates the variance exhibited by the dependent variable. The customary range for R-squared values is 0 to 1. An R-squared of 0 signifies the model's inability to account for the variance within the dependent variable. In contrast, an R-squared of 1 denotes that the regression model offers a comprehensive representation of the variance in the dependent variable. Consequently, higher R-squared values closer to 1 signify an improved model fit. To calculate R², a series of steps is undertaken:

1) Compute the sum of squares of the discrepancies between the predicted values $\hat{y}$ generated by the model and the actual observed values y, referred to as the Sum of Squares for Error (SSE).
2) Calculate the sum of squares of the discrepancies between the actual observed values y and the mean of the observed data $\overline{y}$, denoted as the Sum of Squares for Total (SST).
3) Finally, R² is derived using the following Equation (12):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}_i\right)^2} = 1 - \frac{SSE}{SST} \tag{12}$$

In summary, among the three evaluations mentioned above

TABLE IV
VARSEG-TRANS AND OTHER MODELS (% DIFFERENCE)

Datasets	Forecasting Steps	Metrics		
		MSE	MAE	R^2
ETTh1	24	1.52%	-9.95%	0.53%
	48	-3.83%	0.67%	2.24%
	96	-4.35%	-0.44%	2.34%
	192	-3.23%	-0.92%	3.22%
ETTh2	24	-18.87%	-12.99%	0.54%
	48	-21.81%	-14.86%	2.14%
	96	-5.58%	-8.99%	0.83%
	192	-15.66%	-11.96%	2.11%
ETTm1	24	12.00%	-7.97%	3.95%
	48	-8.60%	-5.33%	6.55%
	96	-7.06%	-13.36%	3.00%
	192	-5.68%	7.10%	-0.62%
Weather	24	1.74%	2.89%	0.43%
	48	-6.36%	-2.76%	1.13%
	96	-10.53%	-22.04%	3.58%
	192	-7.98%	-10.12%	1.38%
Electricity	24	2.88%	9.52%	-1.26%
	48	-5.62%	3.47%	1.04%
	96	-6.67%	-16.09%	3.28%
	192	-7.26%	-9.29%	8.65%

metrics, the superior forecasting capability of a model is indicated by lower values of MSE and MAE, along with a higher value of the coefficient of R^2.

D. Result and Analysis

As is shown in Table III, we have presented the predictive results of various models across five different datasets and for five distinct forecasting horizons. Optimal results are denoted in bold, while the second-best results are underlined. In Table IV, we conducted an in-depth analysis of VarSeg-Trans compared to the other models to better represent the experimental outcomes. Positive values in the MSE and MAE columns signify that VarSeg-Trans did not achieve the best performance within the specified dataset and forecast range, with the numerical values quantifying the extent of the deviation from the optimal results. Conversely, negative values indicate that VarSeg-Trans outperformed the second-best result within the designated dataset and forecasting range, with the numerical values reflecting the degree of its superiority. As for the R^2 column, positive values indicate that VarSeg-Trans achieved the best performance within the given dataset and forecast range, with the numerical values measuring the extent of its superiority over the second-best results, while negative values signify that VarSeg-Trans did not attain the optimal results within the designated dataset and forecast range, with the numerical values representing the disparity from the optimal results.

Compared to other models, VarSeg-Trans exhibits a noteworthy advantage on the ETTh1 dataset, with reductions of 2.47% in MSE, 2.66% in MAE, and a 2.08% increase in R^2 across four distinct forecast horizons. On the ETTh2 dataset, VarSeg-Trans outperforms other models by reducing MSE and MAE by 15.48% and 12.2%, respectively, with a 1.41% R^2 improvement. In the case of the ETTm1 dataset, VarSeg-Trans surpasses other models with a 2.33% reduction in MSE, a 4.89% reduction in MAE, and a 3.22% increase in R^2 across four forecast horizons. For the Weather dataset, VarSeg-Trans outperforms other models with MSE and MAE reductions of 5.78% and 8.01%, respectively, alongside a 1.63% increase in R^2 across four forecast horizons. Lastly, on the Electricity dataset, VarSeg-Trans excels by reducing MSE and MAE by 4.17% and 3.1%, respectively, and increasing R^2 by 2.93% across four forecast horizons in comparison to the second-best model.

For brevity, we classify forecast horizons less than or equal to 48 as short-to-medium-term forecasting, while those greater than 48 are categorized as long-term forecasting. Baseline comparisons reveal that LSTM delivers strong performance for 24-step forecasts on specific datasets. However, due to inherent issues with gradient vanishing, its performance lags behind Transformer-based models for forecasts exceeding 24 steps. Reformer's Hashing Attention mechanism effectively mitigates errors for longer forecast horizons. However, it performs poorly for short horizons, resulting in an overall drop in performance due to increased short-to-medium-term forecast errors. Notably, VarSeg-Trans' cut-up mechanism successfully mitigates these issues. Autoformer's Auto-Correlation mechanism yields decent results for datasets with pronounced periodicity but falters when dealing with less cyclic data. Furthermore, it struggles when the period length of data exceeds the forecast horizon since Autoformer fails to identify the data's periodicity. The top-performing model overall, except for VarSeg-Trans, Informer benefits from the ProbSparse Self-Attention mechanism, providing solid performance for short-to-medium-term forecasts. Nevertheless, when it comes to forecast horizons exceeding 48, its mechanism proves ineffective in leveraging neighboring time series data, resulting in performance degradation. In contrast, VarSeg-Trans's cut-up mechanism and improved Attention computation can better harness this aspect, yielding superior results for long-term forecasts.

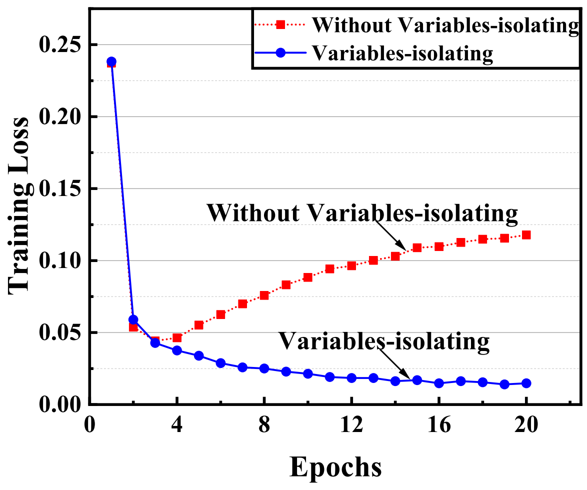

Fig. 5. Mitigating Overfitting with Variable-Isolating Mechanism

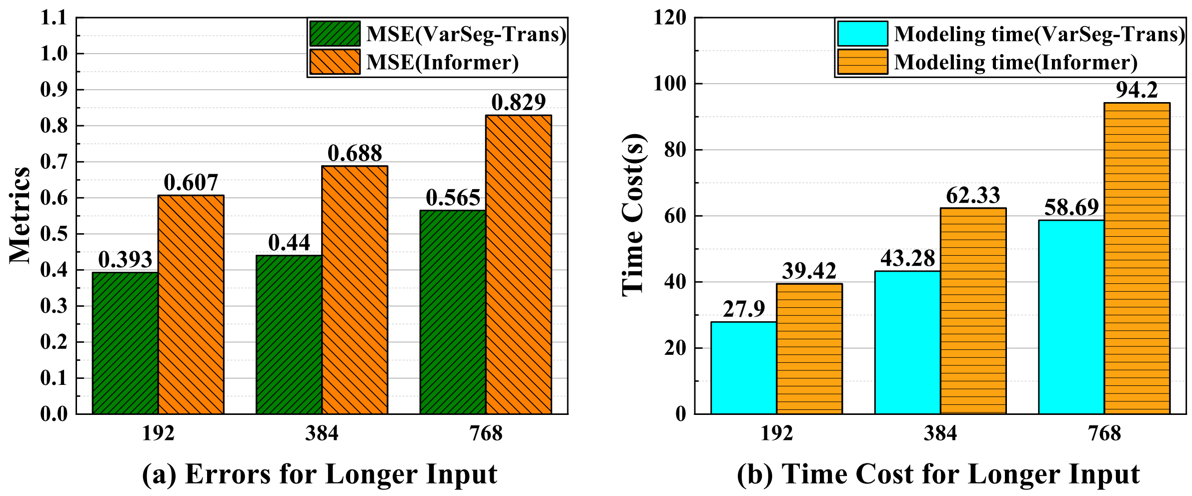

Fig. 6. Comparative Experiments with Longer Inputs

E. Ablation Experiments

To validate the effectiveness of each module and its impact, a series of ablation experiments were conducted on the ETTh1 dataset. Based on the various components of the VarSeg-Trans model, three variants were designed, namely:

1) Seg-Trans: This variant removes the variable-isolation feature from VarSeg-Trans and uses Variable Mixing to handle variable inputs. It aims to test the effectiveness of the variable-isolation mechanism.
2) Var-Trans: In this variant, the removal of the cut-up mechanism transforms VarSeg-Trans, treating each time point as a single input, with the primary objective of thoroughly evaluating the effectiveness of the cut-up mechanism.
3) VarSeg-Inform: In this variant, the Transformer model in VarSeg-Trans is replaced with the Informer model, specifically using Dynamic ProbSparse Attention instead of ProbSparse Attention. The goal is to evaluate the effectiveness of Dynamic ProbSparse Attention in VarSeg-Trans.

These variants were designed to evaluate and validate the impact of individual components of the VarSeg-Trans model on its performance. This approach aims better to understand the model's working principles and advantages. Table V presents the ablation experiments' results on our proposed model and its various sub-models. The Var-Trans model without the cut-up mechanism significantly decreases performance in long-term forecasting scenarios among all sub-models. This corroborates the effectiveness of the cut-up mechanism in significantly enhancing the model's accuracy in long-term prediction tasks. The cut-up mechanism empowers the model to treat multiple time points as a single input, thereby fully leveraging the information from adjacent data points in the time series.

Furthermore, we replaced our improved Transformer model with the Informer, which resulted in the most pronounced performance decline. This indicates the effectiveness of our proposed Dynamic ProbSparse Attention mechanism. Our mechanism balances computational accuracy and time expenditure compared to the original Self-Attention and ProbSparse Attention mechanisms.

The variables-isolating mechanism primarily comes into play during model training. In theory, the Variable-mixing mechanism possesses the capability to discover cross-variable information. However, it may lead to the model learning numerous irrelevant features, thus diminishing its generalization ability and causing overfitting [35]. In Fig. 5, we illustrate the relationship between test values and epochs. We employ the complete training data and plot the results for the first 20 epochs. The results indicate that the variables-isolating mechanism does not lead to rapid overfitting of the data.

Another notable characteristic of the cut-up mechanism is its ability to enhance the model's capacity to handle longer input sequences, thereby furnishing more comprehensive information and significantly improving predictive accuracy. As depicted in Fig. 6(a), we extended the input sequence L, and the VarSeg-Trans model utilizing the cut-up mechanism exhibited a considerably smaller increase in prediction error compared to other Transformer-based models. This outcome suggests that, unlike other Transformer models, VarSeg-Trans can effectively exploit valuable information from longer input sequences. Additionally, this mechanism effectively mitigates the growth in modeling overhead when processing extended input sequences, as illustrated in Fig. 6(b). It is worth noting that the time data provided here is based on specific experimental configurations, and actual time values may vary depending on different settings.

F. Summarize

VarSeg-Trans surpasses other Transformer-based models and commonly used deep learning models in time series forecasting across nearly all metrics. The variables-isolating mechanism effectively reduces model overfitting. The cut-up mechanism enhances the model's ability to analyze more observation values and detect historical relevance while reducing computational costs. Additionally, the Dynamic ProbSparse Attention calculation mechanism improves the calculation accuracy of ProbSparse Attention, and the time overhead is almost unchanged.

VI. Conclusion

This paper investigates the primary challenges in TSF tasks and proposes a novel Transformer-based model to ad-

TABLE V
ABLATION EXPERIMENTS OF VARSEG-TRANS

The original model and Variants	Forecasting Steps	Metrics		
		MSE	MAE	R^2
VarSeg-Trans	24	**0.334**	**0.380**	**0.942**
	48	**0.377**	**0.453**	**0.911**
	96	0.418	**0.449**	0.874
	192	**0.449**	**0.537**	**0.834**
Seg-Trans	24	0.342	0.391	0.938
	48	0.388	0.460	0.902
	96	**0.417**	0.453	0.869
	192	0.462	0.550	0.827
VarSeg-Informer	24	0.347	0.427	0.925
	48	0.393	0.457	0.894
	96	0.420	0.458	0.863
	192	0.578	0.667	0.796
Var-Trans	24	0.351	0.402	0.927
	48	0.400	0.462	0.894
	96	0.431	0.461	0.872
	192	0.502	0.581	0.816

dress them. VarSeg-Trans adopts two effective mechanisms, cut-up and variables-isolating, along with a more accurate attention calculation mechanism. In contrast to prior works, VarSeg-Trans successfully mitigates model overfitting, captures long-term dependencies in time series more effectively, and reduces modeling time. Experimental results on real-world data demonstrate the model's potential for diverse applications and its suitability as a basis for TSF tasks. The mechanisms in VarSeg-Trans can be readily integrated into other models to enhance their ability, paving the way for future breakthroughs.

We anticipate extensive investigations into VarSeg-Trans's performance across diverse real-world scenarios and domains. We express keen interest in delving into potential extensions and adaptations that cater to the challenges of TSF in diverse data types. Additionally, a compelling avenue for future research lies in unraveling the interpretability of the model and examining its capacity to handle uncertainty in time series forecasting tasks effectively.

REFERENCES

[1] R. Barzegar, M. T. Aalami, and J. Adamowski, "Short-term water quality variable prediction using a hybrid CNN–LSTM deep learning model," *Stochastic Environmental Research and Risk Assessment*, vol. 34, no. 2, pp. 415–433, 2020.

[2] N. Jiang, X. Zheng, H. Zheng, and Q. Zheng, "Long Short-Term Memory based PM2.5 Concentration Prediction Method," *Engineering Letters*, vol. 29, no. 2, pp. 765–774, 2021.

[3] J. Wang, H. Li, and H. Lu, "Application of a novel early warning system based on fuzzy time series in urban air quality forecasting in China," *Applied Soft Computing*, vol. 71, pp. 783–799, 2018.

[4] L. BAI, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 17 804–17 815.

[5] B. Yu, Y. Lee, and K. Sohn, "Forecasting road traffic speeds by considering area-wide spatio-temporal dependencies based on a graph convolutional neural network (GCN)," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 189–204, 2020.

[6] T. Zhang and G. Guo, "Graph Attention LSTM: A Spatiotemporal Approach for Traffic Flow Forecasting," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 2, pp. 190–196, 2022.

[7] P. Mondal, L. Shit, and S. Goswami, "Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices," *International Journal of Computer Science, Engineering and Applications*, vol. 4, no. 2, pp. 13–29, 2014.

[8] M. Kumar and M. Thenmozhi, "Forecasting stock index returns using ARIMA-SVM, ARIMA-ANN, and ARIMA-random forest hybrid models," *International Journal of Banking, Accounting and Finance*, vol. 5, no. 3, pp. 284–308, 2014.

[9] J. E. Jarrett and E. Kyper, "ARIMA Modeling with Intervention to Forecast and Analyze Chinese Stock Prices," *International Journal of Engineering Business Management*, vol. 5, no. 3, pp. 17–28, 2011.

[10] M. L. Abdulrahman, K. M. Ibrahim, A. Y. Gital, F. U. Zambuk, B. Ja'afaru, Z. I. Yakubu, and A. Ibrahim, "A Review on Deep Learning with Focus on Deep Recurrent Neural Network for Electricity Forecasting in Residential Building," *Procedia Computer Science*, vol. 193, pp. 141–154, 2021, 10th International Young Scientists Conference in Computational Science, YSC2021, 28 June – 2 July, 2021.

[11] Kong, Weicong and Dong, Zhao Yang and Jia, Youwei and Hill, David J. and Xu, Yan and Zhang, Yuan, "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019.

[12] T. Le, M. T. Vo, B. Vo, E. Hwang, S. Rho, and S. W. Baik, "Improving Electric Energy Consumption Prediction Using CNN and Bi-LSTM," *Applied Sciences*, vol. 9, no. 20, 2019.

[13] G. Sanchez-Torres and I. D. Bolaño, "Support Vector Regression for PM10 Concentration Modeling in Santa Marta Urban Area," *Engineering Letters*, vol. 27, no. 3, pp. 432–440, 2019.

[14] L. Zhang, L. Luo, L. Hu, and M. Sun, "An SVM-Based Classification Model for Migration Prediction of Beijing," *Engineering Letters*, vol. 28, no. 4, pp. 1023–1030, 2020.

[15] H. Zhou, Y. Zhang, L. Yang, Q. Liu, K. Yan, and Y. Du, "Short-Term Photovoltaic Power Forecasting Based on Long Short Term Memory Neural Network and Attention Mechanism," *IEEE Access*, vol. 7, no. 12, pp. 78 063–78 074, 2019.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017, pp. 1082–1093.

[17] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019, pp. 933–944.

[18] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big Bird: Transformers for Longer Sequences," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 17 283–17 297.

[19] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang *et al.*, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021.

[20] H. Zhou, J. Li, S. Zhang, S. Zhang, M. Yan, and H. Xiong, "Expanding the prediction capacity in long sequence time-series forecasting," *Artificial Intelligence*, vol. 318, pp. 103 886–103 901, 2023.

[21] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 419–22 430, 2021.

[22] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 154, pp. 2194–2208, 2021.

[23] L. Schwenke and M. Atzmueller, "Show Me What You're Looking For: Visualizing Abstracted Transformer Attention for Enhancing Their Local Interpretability on Time Series Data," *The International FLAIRS Conference Proceedings*, vol. 34, pp. 3512–3518, Apr. 2021.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012, pp. 84–90.

[25] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456.

[26] T. Kudo, "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates," in *Proceedings of*

the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 66–75.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–789.

[29] D. Yuan and Y. Xu, "Lightweight Vehicle Detection Algorithm Based on Improved YOLOv4," *Engineering Letters*, vol. 29, no. 4, pp. 1544–1551, 2021.

[30] Y. Gao, M. Zhou, and D. N. Metaxas, "UTNet: A Hybrid Transformer Architecture for Medical Image Segmentation," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, M. de Bruijne, P. C. Cattin, S. Cotin, N. Padoy, S. Speidel, Y. Zheng, and C. Essert, Eds. Cham: Springer International Publishing, 2021, pp. 61–71.

[31] A. J. Santoso and Pranowo, "Medical Image Segmentation Using Phase-Field Method based on GPU Parallel Programming," *Engineering Letters*, vol. 30, no. 1, pp. 214–220, 2022.

[32] Y. Li, B. Wang, K. Zhang, Z. Jiang, W. Shi, and W. Liu, "A Novel Automatic Method Based on U-Net for Lung Fields Segmentation," *Engineering Letters*, vol. 30, no. 2, pp. 636–643, 2022.

[33] Q. D. Nguyen, N. M. Phan, and Z. Ivan, "Forecasting time series with long short-term memory networks," *CTU Journal of Innovation and Sustainable Development*, vol. 12, no. 2, pp. 53–59, Jul. 2020.

[34] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International Joint Conference on Neural Networks (IJCNN)*, vol. 10, no. 8, 2017, pp. 1578–1585.

[35] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding Deep Learning (Still) Requires Rethinking Generalization," *Commun. ACM*, vol. 64, no. 3, p. 107–115, Feb. 2021.