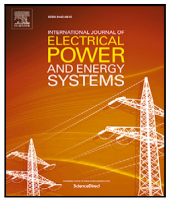
Electrical Power and Energy Systems 155 (2024) 109515

Contents lists available at ScienceDirect

International Journal of Electrical Power and Energy Systems

journal homepage: www.elsevier.com/locate/ijepes



# Interpretable transformer-based model for probabilistic short-term forecasting of residential net load☆

Chongchong Xu, Guo Chen *

*School of Automation, Central South University, Changsha, 410083, China*

ARTICLE INFO

*Keywords:*
Residential net load forecasting
Probabilistic short-term forecasting
Transformer model
Attention mechanism
Interpretable deep learning

ABSTRACT

Short-term residential load forecasting is of great significance for the demand-side energy management of the grid and the home energy management of resident customers. The massive penetration of distributed renewable energy, especially small-scale solar photovoltaic (PV), in the residential sector urgently requires us to move from traditional load forecasting to net load forecasting. In recent years, deep learning techniques that can improve model forecasting performance have developed rapidly in the field of residential load forecasting. However, the opaque nature of deep learning makes its practical application very difficult. This paper proposes a Transformer-based probabilistic residential net load forecasting method that utilizes quantile regression to quantify uncertainty in future load demand. Meanwhile, to improve the interpretability of deep learning model, local variable selection network is developed to automatically select relevant features and provide feature-level explanations. Additionally, interpretable sparse self-attention mechanism is proposed to extract long-term temporal dependencies. Numerical experiments are carried out with data from real household smart meters. The results show that the proposed model outperforms other state-of-the-art forecasting models. In terms of point forecasting, compared with the most common deep time series forecasting model LSTM, the proposed model decreases by 21.3%, 27.3% and 20.9% On three point forecasting performance metrics. Compared with Vanilla Transformer, the proposed InterFormer decreases by 9.9%, 10.5% and 9.0% On three point forecasting performance metrics. In terms of probabilistic forecasting, compared with LSTM and Vanilla Transformer, the average pinball loss of the proposed model decreases by 26.2% and 17.0%, respectively. Additionally, and most importantly, our model provides users with explanations in terms of feature importance and temporal patterns.

## 1. Introduction

With the development and application of emerging Internet of Things technologies, the degree of intelligence and decentralization of energy management has gradually increased [1]. Short-term residential load forecasting is very significant for both the demand-side energy management of the grid and the home energy management of users [2,3]. Based on short-term load forecasts, system operators identify target customers and estimate load flexibility to optimize load shifting. Home energy management systems rely on short-term load forecast results to perform load monitoring and plan energy usage. On the other hand, residential load forecasting plays an important role in the operation of the local energy market as a reference for peer-to-peer energy trading [4].

Distributed renewable energy, especially small-scale solar PV power generation, is rapidly gaining popularity at the residential sector. Generally, only one two-way smart meter is installed in each household to record the difference between power consumption and photovoltaic power generation, i.e., the net load [5]. Since the data obtained by the electricity meter monitoring has been transformed from the actual electricity consumption to the net load, it is urgent to shift research from traditional load forecasting to residential net load forecasting.

Compared with the aggregated load, the power load of a single house is highly correlated with the consumption pattern of the residents, which makes the load profile show great uncertainty [6]. The connection of PV power generation behind the meter makes the net load profile more irregular relative to the actual consumption of users and more relevant to meteorological factors. Most of the existing methods predict the expected value of future loads [7]. However, it is difficult for such point forecasting models to model the uncertainty of the future and provide the possibility for reliable decision-making. Probabilistic forecasting provides a solution that can produce forecasts in the

☆ This work is supported by the National Natural Science Foundation of China under Grant 62073344.
* Corresponding author.
*E-mail address:* guo.chen@csu.edu.cn (G. Chen).

https://doi.org/10.1016/j.ijepes.2023.109515
Received 17 December 2022; Received in revised form 26 August 2023; Accepted 17 September 2023
Available online 14 October 2023

form of quantiles, intervals, or probability density functions, suitable for quantifying uncertainty in net load [8]. Probabilistic forecasting results provide the basis for many uncertain decision-making processes, such as probabilistic energy dispatch optimization [9] and optimal bidding in electricity markets [10]. These decision-making processes utilize uncertain optimization methods such as robust optimization or stochastic programming to balance reliability and economy [11]. Therefore, it is necessary to establish a probabilistic forecasting model for residential net load forecasting.

Accurately predicting residential net load is a challenging problem. In recent years, deep learning has made great progress in solving the problem of time series forecasting of short-term residential loads. [12]. Recurrent Neural Network (RNN) is a network specially designed to deal with serial correlations. The development of Long Short-Term Memory (LSTM) network has enabled RNN to handle long-term dependencies. In [6], a model based on a deep LSTM network was built to predict the single-family residential load with high volatility. Ref. [13] proposed a residential net load prediction model based on multiple-input single-output LSTM. The historical net loads of multiple other households were used as input to generate more accurate predictions for the target household. In addition, the model adopted an online training method to resist the sudden change of the load profile. Ref. [7] combined Bayesian deep learning with LSTM for probabilistic prediction of residential net load at aggregate level. It leveraged the efficient uncertainty capture capability of Bayesian deep learning and the powerful temporal dependencies extraction capability of LSTM. The encoder–decoder architecture based on the RNN model was also applied to residential load forecasting to achieve multi-step forecasting [14]. In addition, Another representative of deep learning algorithms, Convolutional Neural Network (CNN), has also been applied to short-term residential load forecasting. Convolutional filters in CNN can extract local and periodic patterns in sequences and learn nonlinear features from data for prediction. In [15], two CNNs were constructed to extract nonlinear load features and nonlinear load temperature features, respectively. The combination of CNN and RNN is also a research direction, which can significantly improve the feature extraction ability of the model. Ref. [16] proposed a CNN-attention-bidirectional LSTM based model for residential load forecasting. An attention mechanism was applied to assign different weights to the hidden layer outputs to select salient hidden layer information.

Recently, Transformer model has made great progress in time series forecasting, especially multi-step forecasting [17,18]. Transformer model relying only on the self-attention mechanism allows the model to access any part of the input regardless of their distance in the sequence, making it easier to capture long-range temporal dependencies. In addition, the self-attention mechanism allows the Transformer model to avoid recurrent structure and be trained in parallel. Ref. [19] applied Transformer to the load forecasting problem by adding N-space transform and context information extraction module to vanilla Transformer. The model achieved better prediction performance with longer prediction horizon and smaller input data volumes. In [20], a multi-decoder Transformer was proposed to implement a multi-task learning for simultaneous prediction of different types of loads in an integrated energy system. The proposed multi-task Transformer model extracts coupled information through one encoder and predicts different types of loads through different decoders. Ref. [21] proposed a short-term load forecasting model that combines the Transformer model and an improved empirical mode decomposition algorithm. Modal decomposition techniques that decomposes load sequence into multiple frequency subsequences can reduce the impact of noise on prediction performance.

In general, most of the load forecasting models described above are "darker" models generated by complex nonlinear interactions of a large number of parameters. They mainly consider prediction performance, but ignore model interpretability. End users and even developers are not clear about how the model makes predictions, which leads to low model credibility. Several attention-based methods, including Transformer models, have been proposed to improve interpretability. But time series forecasting includes many different types of input features. A single temporal attention can discriminate correlations at time steps, but cannot increase insight into relevant features [22]. Therefore, more attention needs to be paid to feature-level interpretability when building models.

In this paper, we propose an interpretable Transformer-based net load forecasting model named InterFormer which can simultaneously predict the net load of multiple time steps in the future and achieve the interpretability of the forecasting results. First, in order to optimally utilize the information carried by different variables, we design a local variable selection network to extract effective information from the original input. The information from the variable selection network is then fed into a generative Transformer that efficiently captures temporal dependencies through an interpretable sparse self-attention mechanism. The model then generates probabilistic predictions for future multiple time steps through quantile regression.

As stated above, the contributions of this paper can be summarized as follows.

(1) Compared with existing methods, a local variable selection network is established that automatically selects relevant features among the available variables in different windows. This network provides insight into which variables drive predictions, giving instance-wise feature importance.
(2) An interpretable sparse self-attention mechanism is adopted in InterFormer to model the long-term dependencies of all input time steps. The self-attention mechanism captures persistent temporal patterns in the data by identifying relevant time steps in multi-step predictions.
(3) A generative Transformer prediction module that abandons the Encoder–Decoder architecture is proposed. The signaling path between the prediction and the input is shortened, making it easier for the model to take into account historical observations. In addition, probabilistic prediction is extended with quantile regression.

The rest of this paper is organized as follows. The probabilistic forecasting problem of residential net load is presented in Section 2. The Transformer-based interpretable prediction model is described in detail in Section 3. Section 4 presents a case study of residential net load forecasting and discusses the interpretability of the model. Section 5 draws a conclusion.

## 2. Problem formulation

In this section, we give the problem statement of the probabilistic short-term forecasting of residential net load. The net load of an individual household not only depends on consumption patterns of customers, but also is highly correlated with environmental factors. In order to improve prediction accuracy, we make use of as many available features as possible, including historical information (historical load, weather, etc.), and known information about the future (calendar information, weather forecasts, etc.).

Quantile regression is an implementation of probabilistic forecasting that estimates the value of a specific quantile without considering the specific parametric distribution [23]. Quantiles can flexibly quantify the uncertainty of future net load and improve grid risk management. Therefore, we let the model output the net load prediction results at different quantile levels.

Let $\mathbf{y}_{i,1:t_0}$ denote the net load observations of the customer $i$, where $\mathbf{y}_{i,1:t_0} := [y_{i,1}, y_{i,2}, \dots, y_{i,t_0}]$ and $t_0$ is the input sequence length. In addition to historical observations of net load, other dynamic time-dependent multidimensional covariates $\mathbf{X}_{i,1:t_0+\tau}$ can be accessed, where $\tau$ is the prediction horizon. These covariates can be divided into two categories, i.e., observation covariates $\mathbf{Z}_{i,1:t_0}$ whose values are collected

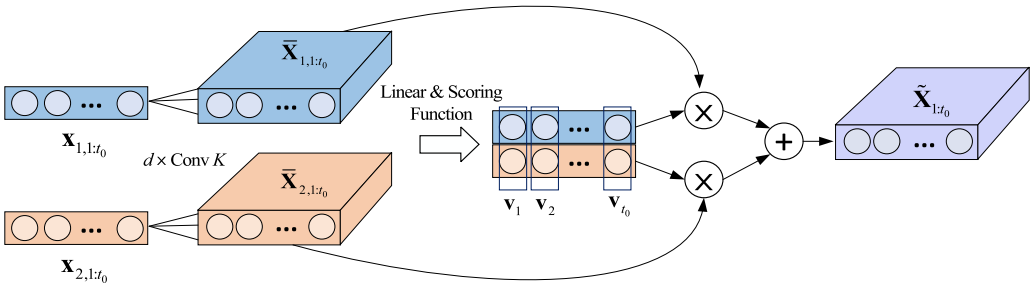



**Fig. 1.** An example of a local variable selection network with two variable input sequences in the condition window.

sequentially over time, and known covariates $\mathbf{C}_{i,1:t_0+\tau}$ which can be obtained before prediction, such as calendar variables. For clarity, the time range $[1, t_0+\tau]$ is divided into condition window $[1, t_0]$ and prediction window $[t_0+1, t_0+\tau]$. For rolling forecasting with a fixed window length, Our task is predicting the net load quantiles for the next $\tau$ time steps given the history observations and all covariates:

$$\hat{\mathbf{Y}}_{t_0+1:t_0+\tau} = F(\mathbf{y}_{1:t_0}, \mathbf{Z}_{1:t_0}, \mathbf{C}_{1:t_0+\tau}) \tag{1}$$

where $\hat{\mathbf{Y}}_{t_0+1:t_0+\tau} := [\hat{\mathbf{y}}_{t_0+1}, \hat{\mathbf{y}}_{t_0+2}, \dots, \hat{\mathbf{y}}_{t_0+\tau}]$, $\hat{\mathbf{y}}_t = \{\hat{y}_t^q\}_{q\in Q}$ for target quantile set $Q$ at time step $t \in [t_0+1, t_0+\tau]$, $F(\cdot)$ is a prediction model. Note that we omit the customer ID $i$ in the rest of the paper for simplicity.

## 3. Interpretable transformer-based model

### 3.1. Local variable selection network

Typically, we directly combine different exogenous variables with historical observations of net load as input to the model. But their actual correlation to net load forecasts is often unknown, making deep learning model difficult to interpret for customers. To this end, we develop a new local variable selection network to integrate all input features. This network first learns the local context information of a single variable through a convolution operation, and then learns weight parameters through a scoring function to represent the variable importance. In addition to giving an explanation of the importance of variables, it can also improve model performance by increasing the weights of salient features.

Whether a point in a sequence is an anomaly, a change, or part of a pattern is very dependent on its local context [24]. The amount of information carried by a variable at a certain time is also highly related to its local environment. To be fully aware of the local context, we apply a 1D causal convolution [25] to each individual input variable. Specifically, we have 1 filter $C_n \in R^{d\times K}$ of kernel size $K$ and stride 1 with same padding for each input variable. For the $n$th variable $\mathbf{x}_n$ of input $\mathbf{X} \in R^{t_0\times N}$ and the $j$th convolution kernel $C_{n,j} \in R^K$ in the condition window, where $N$ is the number of input variables, we do the following transformation:

$$\bar{x}_{n,j}[m] = (\mathbf{x}_n * C_{n,j})[m] = \sum_{k=0}^{K-1} C_{n,j,k} \times x_{n,m-k} \tag{2}$$

By using convolution operation, we get the local univariate state $\bar{\mathbf{x}}_{n,t} := [\bar{x}_{n,1,t}, \bar{x}_{n,2,t}, \dots, \bar{x}_{n,d,t}] \in R^d$ at time step $t$, and the local univariate state matrix of condition window $\bar{\mathbf{X}}_{n,1:t_0} := [\bar{\mathbf{x}}_{n,1}, \bar{\mathbf{x}}_{n,2}, \dots, \bar{\mathbf{x}}_{n,t_0}] \in R^{t_0\times d}$. The method presented here is for condition window, but the same calculations apply to the prediction window. Using the local contextual information of variables, such as variable shape, for variable selection helps the model to extract effective variable information and improve prediction performance.

Next, all the local univariate states are combined and input into a variable integration network. Combining all the variables helps to identify the significance of the variables. Although univariate analysis can also find correlations between input variables, it cannot discern whether variables are redundant [26]. Specifically, we transform the variable state matrix $\bar{\mathbf{X}}_t := [\bar{\mathbf{x}}_{1,t}, \bar{\mathbf{x}}_{2,t}, \dots, \bar{\mathbf{x}}_{N,t}] \in R^{N\times d}$ of time step $t$ into an $N$-dimensional vector $\hat{\mathbf{x}}_t \in R^N$ through a linear transformation, and then normalize it through $\alpha-\text{entmax}$ scoring function:

$$\hat{\mathbf{x}}_t = \bar{\mathbf{X}}_t \mathbf{w}_s + \mathbf{b}_s \tag{3}$$

$$\mathbf{v}_t = \alpha-\text{entmax}(\hat{\mathbf{x}}_t) \tag{4}$$

where $\mathbf{v}_t \in R^N$ is the feature weight vector, $\mathbf{w}_s \in R^d$ and $\mathbf{b}_s \in R^N$ are parameters optimized during the training. Traditionally, the scoring function is computed via the softmax function. The softmax function never assigns zero probability to the input, which means that all features affect the prediction [27]. In addition, this inevitably means less attention to important variables because all the scores add up to 1, which is likely to hurt the performance of the prediction model and also increase the difficulty of interpretation. $\alpha-\text{entmax}$ [28] is a more optimal family of transformations that flexibly map inputs to sparse probabilities. It retains only the scores of important variables when calculating, excluding irrelevant input variables. The calculation method of $\alpha-\text{entmax}$ is as follows:

$$\alpha-\text{entmax}(\hat{\mathbf{x}}_t) = \text{ReLU}\left((\alpha-1)\hat{\mathbf{x}}_t - p\mathbf{1}\right)^{1/\alpha-1} \tag{5}$$

where $p$ is the Lagrange multiplier and $\mathbf{1}$ is the all-one vector. When $\alpha = 1$, $\alpha-\text{entmax}$ is equivalent to the softmax function. For any value of $\alpha > 1$, the transformation returns a sparse probability vector, and the larger the value of $\alpha$, the sparser the probability distribution. When $\alpha = 2$, $\alpha-\text{entmax}$ is equivalent to the sparsemax function [29]. We set $\alpha$ to 1.5, which works well in practical applications with specialized fast algorithms.

Finally, The original variable state $\bar{x}_{n,t}$ are weighted in the following:

$$\tilde{\mathbf{x}}_t = \sum_{n=1}^{N} v_{n,t}\bar{\mathbf{x}}_{n,t} \tag{6}$$

where $v_{n,t}$ is the $n$th element of vector $\mathbf{v}_t$. For condition and prediction window, we get $\tilde{\mathbf{X}}_{1:t_0} \in R^{t_0\times d}$ and $\tilde{\mathbf{X}}_{t_0+1:t_0+\tau} \in R^{\tau\times d}$ through two different local variable selection networks, respectively. The weighted feature vector $\tilde{\mathbf{x}}_t \in R^d$ at each time step naturally pays attention to the significant variables and filters out irrelevant variables before being fed into the predictor. The use of local variable selection network helps the prediction model to have better performance when dealing with more input features. Fig. 1 provides an example of a local variable selection network with two variable input sequences in the condition window.

### 3.2. Interpretable sparse self-attention mechanism

Attention mechanism is a commonly used method in time series forecasting [30]. It is generally applied in an encoder–decoder (seq2seq) framework to spontaneously identify time steps in the encoder sequence that are important to the decoder output [31]. Self-attention mechanism was first proposed for Transformer in [32]. Self-attention is not an attention mechanism between encoder and decoder,

but an attention mechanism that occurs between internal elements of an input. Self-attention mechanism enables Transformer to fully explore the internal relationship between inputs and capture long-term or short-term temporal dependencies. For time series forecasting, self-attention mechanism can help prediction models pay attention to important steps in the input sequence, regardless of the length of the input time steps. The use of self-attention can lead to better performance of the model in long sequence prediction problem.

Generally speaking, the self-attention mechanism is described by the scaled dot-product attention function [20]. In the self-attention layer of the Vanilla Transformer, softmax is used to output the normalized attention scores after the dot-product operation [17]. But softmax always leads to dense attention weights, which is computationally disadvantageous and not conducive to identify temporal patterns [33]. Considering that only a few historical time steps are relevant for prediction, we continue to use $\alpha-\text{entmax}$ to achieve sparse self-attention, and the calculation method is as follows:

$$\begin{cases}\mathbf{Q}=\mathbf{X}_{in}\mathbf{W}_Q\\ \mathbf{K}=\mathbf{X}_{in}\mathbf{W}_K\\ \mathbf{V}=\mathbf{X}_{in}\mathbf{W}_V\end{cases} \tag{7}$$

$$\begin{aligned}\text{Att}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \pi(\mathbf{Q},\mathbf{K})\mathbf{V}\\ &= \alpha-\text{entmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}\end{aligned} \tag{8}$$

where $\mathbf{X}_{in}$ is the input from the previous layer, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are the query, key and value matrices, $\mathbf{W}_Q\in R^{d\times d_k}$, $\mathbf{W}_K\in R^{d\times d_k}$ and $\mathbf{W}_V\in R^{d\times d_v}$ are the corresponding linear transformation matrices, $\pi(\cdot)$ is a normalization function. It is worth noting that the query, key and value are all transformed from the same input, so the attention here is called self-attention.

To improve the feature extraction ability, multi-head attention was proposed in [32]. Multi-head attention is to map $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ to different representation subspaces through linear transformation, and perform multiple attention operations. The multi-head attention results are finally concatenated together directly and are linearly transformed into the attention output. Given the attention matrices of different subspaces, it is difficult to obtain a reasonable interpretation of temporal patterns. Therefore, we adopt the average aggregation method of multi-head weights proposed in [34] to obtain multi-head attention output $\mathbf{H}$:

$$\begin{aligned}\mathbf{H} &= \pi_H(\mathbf{Q},\mathbf{K})\mathbf{V}\\ &= \left(\frac{1}{H}\sum_{h=1}^{H}\pi\left(\mathbf{Q}^h,\mathbf{K}^h\right)\right)\mathbf{V}\\ &= \left(\frac{1}{H}\sum_{h=1}^{H}\pi\left(\mathbf{X}_{in}\mathbf{W}_Q^h,\mathbf{X}_{in}\mathbf{W}_K^h\right)\right)\mathbf{X}_{in}\mathbf{W}_V\\ &= \frac{1}{H}\sum_{h=1}^{H}\text{Att}\left(\mathbf{X}_{in}\mathbf{W}_Q^h,\mathbf{X}_{in}\mathbf{W}_K^h,\mathbf{X}_{in}\mathbf{W}_V\right)\end{aligned} \tag{9}$$

where $H$ is the number of heads, $\mathbf{Q}^h$ and $\mathbf{K}^h$ are the query and key for the $h$th head, $\mathbf{W}_Q^h$ and $\mathbf{W}_k^h$ are the corresponding transformation matrices about the $h$th head. Next, We convert the dimension of $\mathbf{H}$ to the dimension of the predictor by a linear transformation:

$$\tilde{\mathbf{H}}=\mathbf{H}\mathbf{W}_H \tag{10}$$

where $\mathbf{W}_H\in R^{d_v\times d}$ is weight parameter. The average aggregation function $\pi_H(\mathbf{Q},\mathbf{K})$ enables each head to learn different temporal patterns while paying attention to the input in the same space. Fig. 2 shows how the interpretable sparse multi-head attention is calculated.

### 3.3. Generative prediction architecture based on transformer

There are three main strategies for multi-step forecasting of time series, namely direct strategy, recursive strategy and multiple-input multiple-output (MIMO) strategy [35]. The direct strategy builds a model for each prediction time step, i.e., $\tau$ models are used to forecast the net load in the future $\tau$ steps respectively. The direct strategy require more computational time and space. Furthermore, the model does not take into account the position information between time steps, which results in discontinuous prediction profiles and poor model performance [36]. The recursive strategy iteratively generates successive predictions based on previous predictions, like the vanilla Transformer. Most recursive models employ the teacher forcing strategy [37] that assumes the previous target values is known during training. And during inference, the previous predictions will be used as the model input for the next prediction. Differences in training and inference may cause errors to accumulate, as small errors that are not important in training will gradually magnify during inference [17]. The MIMO strategy generates the output sequence through a forward process, which can avoid the error accumulation problem and can better capture the temporal dependencies among the input sequences. To implement MIMO strategy using Transformer, a generative encoder–decoder architecture can be employed, as described in [17,18]. The encoder converts the input sequence into context vectors, and the decoder directly generates multi-step predictions based on the context vectors and the input of the prediction window.

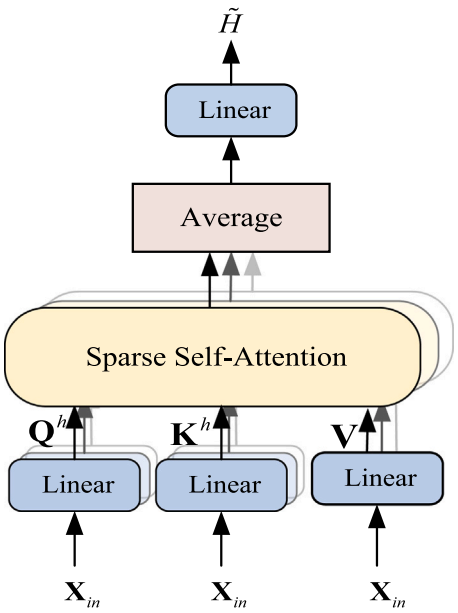



**Fig. 2.** Interpretable sparse multi-head attention.

In this study, we do not adopt encoder–decoder architecture, but use the self-attention layer of decoder in vanilla Transformer directly after the local variable selection network. In general, an encoder–decoder architecture is adopted to handle the difference in the number of input features for condition and prediction windows. However, we have transformed all inputs through a local variable selection network into a weighted feature vector $\tilde{\mathbf{x}}_t\in R^d$, which is consistent with the prediction model in terms of dimension. Another reason to choose the direct generative architecture is that the output can more directly access the historical input in the condition window. The historical input in autoregressive forecasting usually contains the information most relevant to the output. Therefore, the direct generative architecture can enable more efficient back-propagation of errors during training. Fig. 3 shows the architecture of our proposed interpretable multi-step probabilistic net load forecasting model-InterFormer.

For InterFormer, the generative Transformer module consists of multiple layers stacked. Sequence masking [24] is applied to sparse multi-head attention layer to preserve causal information flow. After

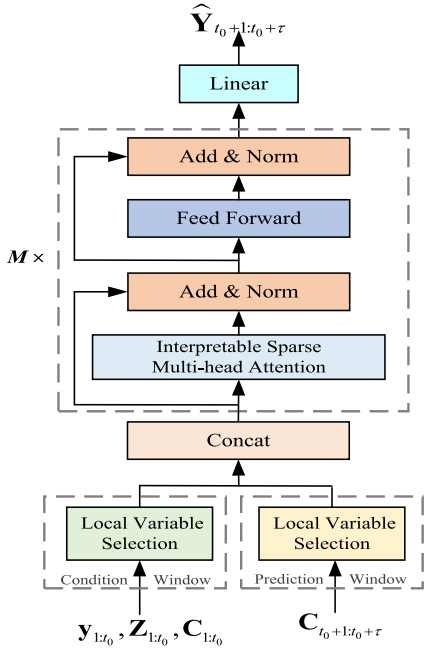



**Fig. 3.** The prediction architecture of InterFormer.

the interpretable sparse multi-head attention layer, a feedforward neural network layer is added to enhance the nonlinear fitting ability of the model. This layer is composed of two linear projections and an activation function, which is calculated as follows:

$$\mathrm{FFN}(\mathbf{x}) = \max\left(0, \mathbf{x}\mathbf{W}_{f_1} + \mathbf{b}_{f_1}\right)\mathbf{W}_{f_2} + \mathbf{b}_{f_2} \tag{11}$$

where $\mathbf{W}_{f_1} \in R^{d\times d_f}$ and $\mathbf{W}_{f_2} \in R^{d_f\times d}$ are the weight parameters, $\mathbf{b}_{f_1} \in R^{d_f}$ and $\mathbf{b}_{f_2} \in R^{d}$ are the corresponding biases. One detail to note is that each sublayer (sparse self-attention layer and feedforward neural network layer) has a residual connection and a layer normalization operation, which is computed as:

$$\mathrm{SubLayerOutput} = \mathrm{LayerNorm}(\mathbf{X} + \mathrm{SubLayer}(\mathbf{X})) \tag{12}$$

Residual connection can alleviate the vanishing gradient problem. Layer normalization converts the input data into normalized data with a mean of 0 and a variance of 1, which can speed up the convergence of the model.

Finally, the prediction model utilizes the output $\hat{\mathbf{H}}_{t_0+1:t_0+\tau} := [\hat{\mathbf{h}}_{t_0+1}, \hat{\mathbf{h}}_{t_0+2}, \ldots, \hat{\mathbf{h}}_{t_0+\tau}]$ of the generative Transformer module and generates prediction intervals by quantile regression in the prediction window. At each time step of the prediction window, the model predicts the $q$-quantile of the net load by:

$$\hat{y}_t^q = \mathbf{W}_y^q \hat{\mathbf{h}}_t + b_y^q \tag{13}$$

where $\mathbf{W}_y^q$ and $b_y^q$ are the parameters for the $q$-quantile prediction. The quantiles are generated in the form of point prediction, which makes no assumption about the form of the target distribution and therefore have more flexibility.

### 3.4. Model interpretability

Feature importance is the most commonly used form of model explanation, and it quantifies the relationship between predictions and input features. In general, the impact of input features on net load forecasting is very complex, and different variables contribute differently to the forecast results. Analyzing the contribution of features such as calendar variables and weather variables to the forecast results can help users understand data-driven deep learning models more thoroughly. We extract general insights about feature importance by analyzing feature weight vectors $\mathbf{v}_t$ in local variable selection networks. Specifically, we aggregate $\mathbf{v}_t$ for the condition and prediction windows of each sample in the training set separately. In the condition window, the global feature importance can be obtained by:

$$I_C^n = \frac{\sum_{s=1}^{S}\sum_{t=1}^{t_0} v_{n,t,s}}{S\times t_0} \tag{14}$$

where $I_C^n$ is the global feature importance of the $n$th variable in the condition window and $\sum_{n=1}^{N} I_C^n = 1$, $v_{n,t,s}$ is the feature importance of the $n$th variable of the $s$th sample at time step $t$, $S$ is the number of samples in the training set. Likewise, the global feature importance of the $n$th variable in the prediction window is:

$$I_P^n = \frac{\sum_{s=1}^{S}\sum_{t=t_0+1}^{t_0+\tau} v_{n,t,s}}{S\times t_0} \tag{15}$$

Analysis of self-attention Weights can help us extract temporal patterns and understand temporal dependencies in net load sequence. Model developers can also use the acquired knowledge about temporal patterns to do further feature engineering to improve model performance. In our proposed model, the attention weight matrix $\pi_H(\mathbf{Q},\mathbf{K})$ in the sparse self-attention layer of the generative Tansformer module gives insight into how well the model utilizes information from different time steps. For the $(i,t)$th element $\pi_{i,t}$ $(i<t)$ of $\pi_H(\mathbf{Q},\mathbf{K})$, its value can be interpreted as the probability that the information input at time step $i$ is aligned with the input at time step $t$, which directly describes the dependencies of the model on inputs at different time steps. For the $m$th sparse self-attention layer, we measure the contribution of the output of the previous layer at past time steps to the output of this layer at time step $t$ by averaging all $\pi_{i,t,m}$ over all training set samples:

$$T_{i,t,m} = \frac{\sum_{s=1}^{S}\pi_{i,t,m,s}}{S} \tag{16}$$

where $T_{i,t,m}$ $(t\in[t_0+1:t+t_0])$ is the temporal importance of time step $i$ to time step $t$ of the $m$th sparse self-attention layer and $\sum_{i=1}^{t} T_{i,t,m} = 1$.

## 4. Case study

### 4.1. Data description and preparation

To evaluate the effectiveness of the model, we conduct a case study on real-world dataset. The data was collected from 28 houses in the Western Victoria region of Australia, and each house is equipped with small solar power generation equipment. The time range of load data is from January 1, 2017 to December 31, 2018, with a time resolution of 30 min. Four calendar variables of month, day-of-week, hour and holiday are considered as input. The corresponding half-hour weather records in the dataset are collected from the nearest weather station, including temperature, relative humidity, dew point, wind speed, sky condition, visibility, precipitation, and direct solar exposure, etc., a total of 8 real-valued variables. Each household in this dataset has only one meter installed to record the net load.

Calendar variables and weather forecasts are future known variables that are used as input for the prediction window. The observed weather data is converted into forecast weather results under approximate conditions for training, but in actual implementation, the weather forecast value needs to be used instead. In addition, the corresponding real-valued variables for each household are scaled to $[0,1]$ to deal with magnitude differences in the data.

### 4.2. Baseline methods and training procedure

We compared InterFormer with multiple multi-step forecasting methods.

(1) LSTM: Long short-term memory neural network. The model generates all quantile forecasts for the next day directly in the last step of the condition window.
(2) Seq2seq: An encoder–decoder model consisting of two LSTMs. The encoder LSTM encodes the history sequence into a context vector. The decoder LSTM uses this context vector as an initial state and extracts the input of the prediction window step by step to generate predictions. The model does not use the prediction from the previous step as input to the next step, which prevents error accumulation.
(3) TCN: Temporal convolutional neural network [25]. The model also has an encoder–decoder structure. The encoder consists of stacked residual blocks based on dilated causal convolutions. The encoder extracts temporal dependencies through convolutions. The decoder is a dense layer based residual network. The decoder produces predictions using the future covariates of the prediction window and the output of the encoder. To make the receptive field of the encoder cover the history sequence of $t_0$ time steps, TCN needs at least $N_c = \lceil log_2(\frac{t_0}{K_c-1} + 1)\rceil$ convolutional layers, where $K_c$ is the size of the convolution kernel.
(4) Vanilla Transformer: The prediction architecture proposed in [32]. The decoder iteratively produces predictions by combining the predictions from the previous step and the input at this time step.
(5) Generative Transformer: Transformer with encoder–decoder structure. The decoder only uses the input of this time step to directly generate the predictions, avoiding error accumulation.

In this study, we use the loads from the previous week to predict the net loads for the next day, i.e., the step sizes of the condition window and the prediction window are 336 and 48, respectively. The quantiles of $q \in Q = \{0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.99\}$ at each time step are output as predictions. All experiments are implemented under the Pytorch [38] framework and trained on a workstation with an Intel Core i9-10900X CPU @ 3.7 GHz and 2 RTX 3090 Ti GPUs. The dataset is split into training, validation and test sets with a ratio of 70/15/15. The loss function of training is defined as the mean pinball loss, which is computed by:

$$L(\mathbf{y}, \hat{\mathbf{y}}, Q) = \frac{\sum_{s_i \in \Omega} \sum_{y_t \in s_i} \sum_{q \in Q} P(y_t, \hat{y}_t^q, q)}{S \times \tau \times |Q|} \tag{17}$$

$$P(y_t, \hat{y}_{t,q}, q) = max((q-1)(y_t - \hat{y}_t^q), q(y_t - \hat{y}_t^q)) \tag{18}$$

where $\Omega$ represents the training set, $|Q|$ is the size of quantile set, and $P(\cdot)$ is the pinball loss function.

We use the mini-batch optimization algorithm Adam to train the model and use the gradient clipping technique to avoid large gradients in iterations. Early stopping is implemented on the validation set to prevent overfitting. Hyperparameter optimization is an important issue in model training. We selected the best hyperparameters by random search.

For all models, the learning rate is chosen from $\{0.0001, 0.001, 0.01\}$, the maximum gradient is chosen from $\{0.1, 1, 10\}$, the batch size is set to is chosen from $\{64, 128, 256\}$, and the dropout rate is chosen from $\{0, 0.1, 0.3, 0.5\}$. In addition, the hidden layer size $d$ is chosen from $\{32, 64, 128\}$ for all models. For all networks except TCN, the number of layer stacks $M$ is chosen from $\{1, 2, 4, 8\}$. For all models with self-attention mechanism, the number of heads is selected from $\{1, 4, 8\}$, and the hidden layer size $d_f$ in FFN is set to 4 times the model dimension. For InterFormer, the convolution kernel size $K$ in the local variable selection network is chosen from $\{2, 4, 6\}$. For TCN model, the convolution kernel size $K_c$ in the encoder is set to 6, and the number of stacked layers is set to 7.

**Table 1**
Performance comparison of different methods for point forecasting.

| Methods | RMSE | MAE | MAAPE |
|---|---|---|---|
| LSTM | 0.3511 | 0.2148 | 0.5072 |
| Seq2seq | 0.2897 | 0.1690 | 0.4396 |
| TCN | 0.3021 | 0.1737 | 0.4494 |
| Vanilla transformer | 0.3065 | 0.1745 | 0.4412 |
| Generative transformer | 0.2877 | 0.1636 | 0.4161 |
| InterFormer | 0.2761 | 0.1562 | 0.4013 |

### 4.3. Evaluation criteria

#### 4.3.1. Point forecasting

We use the 0.5th quantile prediction $\hat{y}_t^{0.5}$ as the expected value of point forecasting. Three metrics i.e. root mean squared error (RMSE), mean absolute error (MAE) and mean arctangent absolute percentage error (MAAPE), are used to assess the performance of point forecasting on the test set. They are computed as follows:

$$RMSE = \sqrt{\frac{1}{T'} \sum_{t=1}^{T'} (y_t - \hat{y}_t^{0.5})^2} \tag{19}$$

$$MAE = \frac{1}{T'} \sum_{t=1}^{T'} |y_t - \hat{y}_t^{0.5}| \tag{20}$$

$$MAAPE = \frac{1}{T'} \sum_{t=1}^{T'} \arctan(\frac{|y_t - \hat{y}_t|}{|y_t|}) \times 100\% \tag{21}$$

where $T'$ is the total number of time steps of test set. MAAPE is an improvement measure of mean absolute percent error (MAPE). The net load level of an individual household is usually low, and there are many points where the load is close to zero or even zero. MAPE produces large outliers in this case. MAAPE performs an arctangent transformation on the relative error, which makes it more robust to outliers.

#### 4.3.2. Probabilistic forecasting

The mean pinball score is an integrated metric for assessing the reliability and clarity of quantile predictions and is calculated as follows:

$$Pinball = \frac{1}{|Q| \times (T')} \sum_{t=1}^{T'} \sum_{q \in Q} P(y_t, \hat{y}_t^q, q) \tag{22}$$

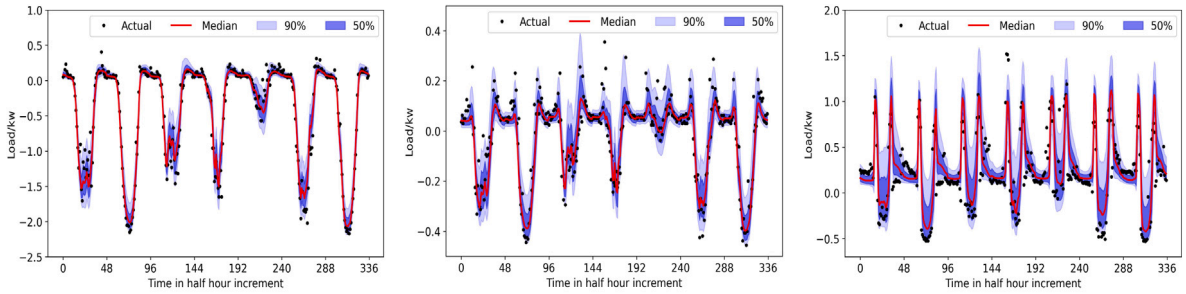
Additionally, the mean Winkler score [39] is also a common measure of quantile prediction performance. For a confidence interval of $(1-\epsilon)\times 100\%$, the Winkler score of one time step is defined as follows:

$$Winkler_t = \begin{cases} \delta_t & L_t \le y_t \le U_t \\ \delta_t + 2\left(L_t - y_t\right)/\epsilon & y_t < L_t \\ \delta_t + 2\left(y_t - U_t\right)/\epsilon & y_t > U_t \end{cases} \tag{23}$$

where $\delta_t = U_t - L_t$ is the width of interval at time step $t$, and $L_t = y_t^{\epsilon/2}$ and $U_t = y_t^{1-\epsilon/2}$ are the lower and upper bounds, respectively. The Winkler score measures the sharpness of the prediction interval when the actual value lies within the prediction interval. Otherwise, a penalty term is added to measure the lack of reliability of the prediction interval. $\epsilon = 0.02, 0.1, 0.2, 0.5$ is used for computing the Winkler score in this study.

### 4.4. Forecasting results

To get reliable prediction results, we train each model 10 times. Table 1 presents the average prediction errors of different model in terms of point forecasting. InterFormer significantly outperforms competing methods including LSTM, Seq2seq, TCN, Vanilla Transformer, and Generative Transformer. Compared with other models, LSTM achieved the




(a) Customer 1 (b) Customer 2 (c) Customer 3

**Fig. 4.** The quantile forecasting results during 7 consecutive days for 3 customers.

**Table 2**
Performance comparison of different methods for probabilistic forecasting.

| Methods | Pinball | Winkler | | | |
|---|---|---|---|---|---|
| | | $\epsilon = 0.02$ | $\epsilon = 0.1$ | $\epsilon = 0.2$ | $\epsilon = 0.5$ |
| LSTM | 0.0516 | 1.9056 | 1.2603 | 1.0141 | 0.6950 |
| Seq2seq | 0.0426 | 1.8118 | 1.1023 | 0.8572 | 0.5612 |
| TCN | 0.0451 | 1.8834 | 1.1685 | 0.9081 | 0.5942 |
| Vanilla transformer | 0.0459 | 2.2131 | 1.2469 | 0.9433 | 0.5876 |
| Generative transformer | 0.0408 | 1.6988 | 1.0471 | 0.8113 | 0.5385 |
| InterFormer | 0.0381 | 1.5476 | 0.9885 | 0.7762 | 0.5126 |

worst prediction results because LSTM cannot use the known information of the future, especially the weather prediction information which has a great influence on the prediction. Compared with LSTM, the proposed InterFormer reduces RMSE, MAE and MAAPE by 21.3%, 27.3% and 20.9%, respectively. It can also be seen from the table that the performance of the Vanilla Transformer based on iterative prediction is not good. Compared with Vanilla Transformer, the proposed InterFormer decrease by 9.9%, 10.5% and 9.0% in terms of RMSE, MAE and MAAPE, respectively. Generative Transformer provides more accurate point forecasting results than the other three methods. Because the self-attention mechanism can easily consider all inputs and extract long-term dependencies from historical data. However, the generative Transformer based on encoder–decoder structure is still inferior to InterFormer. Improvement using our proposed model is evident and demonstrate the effectiveness of directly generative Transformer prediction module combined with local variable selection network.

Next, we perform a probabilistic quantification of future net loads. We calculate the average pinball loss and the Winkler scores when $\epsilon = 0.02, 0.1, 0.2, 0.5$, and the results are shown in Table 2. It can be seen that the trends of the two metrics are similar. Compared with the other 5 methods, the proposed InterFormer can provide the best probability prediction results on all intervals. For example, compared to LSTM and Vanilla Transformer, the average pinball loss of the proposed InterFormer decreased by 26.2% and 17.0%, respectively. This means that InterFormer can effectively capture the uncertainty in net load data and generate accurate quantile prediction intervals. This result is very meaningful since accurate prediction of quantiles with different confidence levels is of great significance to power risk management.

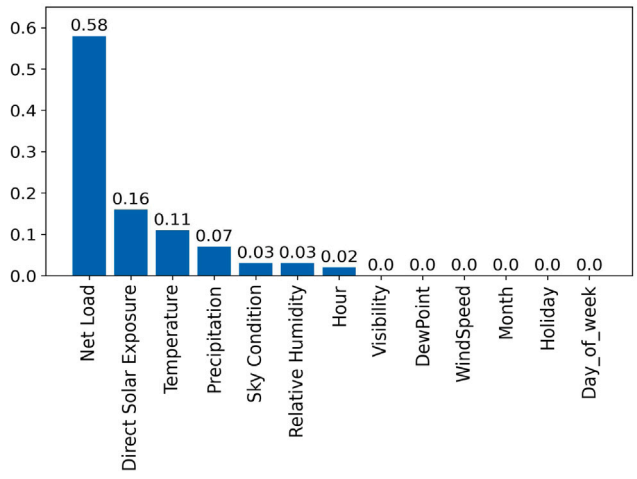
To illustrate the effect of the proposed forecasting model, Fig. 4 visualizes the actual net load observations, point forecasts (median prediction), 50% and 90% prediction intervals in 7 consecutive days for three randomly selected customers. The black dots are actual net load observations, and the red curve denotes the median prediction. Dark and light blue represent the 50% and 90% quantile intervals, respectively. It can be seen that among the three customers, the actual value of the net load is covered with high probability by the 90% prediction interval, except for a few peak loads. Except for the peak load period, the median forecast curve is basically consistent with the actual load. Considering the high volatility of individual residential net load, the proposed model can obtain satisfactory prediction results. Compared with the other two customers, customer 1 has a larger power generation, and its load consumption is much smaller than the power generation, so the change of its peak load consumption is not obvious in the figure. However, the load consumption of the other two customers has obvious peaks in the morning and evening, corresponding to the typical time of going to work and resting respectively. During peak load hours, customer load demand is more uncertain and harder to predict. The peak load consumption of customer 3 is relatively higher, which reflects the different lifestyles of different customers. It is worth mentioning that the solar power generation varies greatly from day to day, especially on the fifth day when the three household customers generated very little power. This was caused by different weather conditions. The fifth day was a rainy day, and the amount of solar radiation received by the power generation equipment was very small. Our proposed model can effectively use the weather forecast information to capture the variation of power generation, thereby improving the prediction performance.

### 4.5. Interpretability analysis

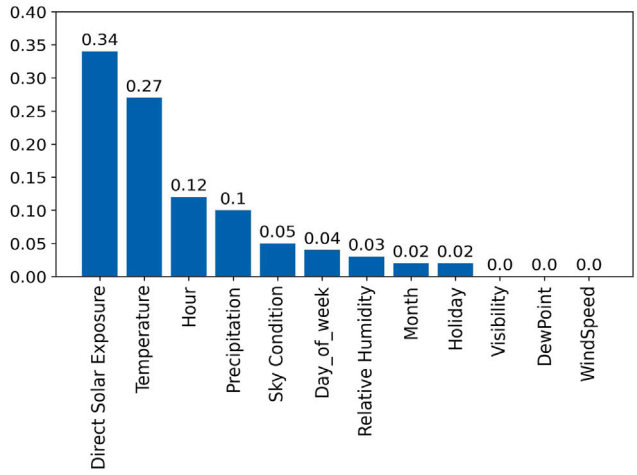
In this section, we analyze how InterFormer interprets predictions. First, we focus on the feature importance learned by the local variable selection network during training. Fig. 5 shows the global feature importance computed by InterFormer. The horizontal axis represents the input variable, and the vertical axis is the global feature importance score. In the condition window, as expected, past net load observation is the most relevant, accounting for 0.58. The historical observation of net load is the most direct reference for forecasting, and is therefore the most important input variable. In addition, weather variables such as direct solar exposure and temperature also contribute to the model predictions, accounting for 0.16 and 0.11 respectively. In fact, they allow the model to make assumptions about the current environmental conditions. In contrast, calendar variables have little value in condition window.

In the prediction window, the weather forecasts and calendar variables are known inputs. Of these, direct solar exposure (contribution 0.34) is the most relevant input, which directly drives the generation of home solar equipment. Temperature (contribution 0.27) is the second relevant variable, which can affect the load consumption of customers (for example, higher or lower temperatures may cause customers to turn on the air conditioning system). Therefore, obtaining accurate weather forecasts is very important for net load forecasting. Furthermore, hour (contribution 0.12) is the third most important variable, which reflects the daily periodicity of the net load. It can also be seen from Fig. 5 that the model only extracts information from a subset of the input variables to produce predictions, which is what $\alpha - \text{entmax}$ does. $\alpha-\text{entmax}$ suppresses irrelevant variables so that they do not affect model predictions.

Attention weight patterns can reveal the most important time steps for making predictions. InterFormer learns these temporal patterns

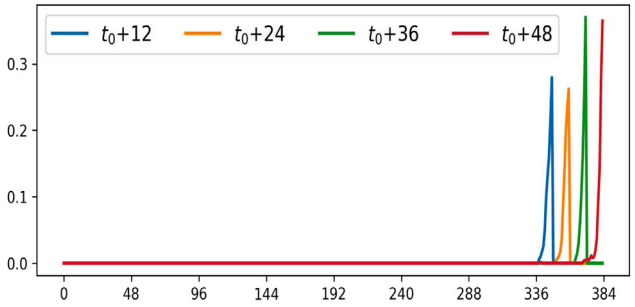



(a) Condition Window

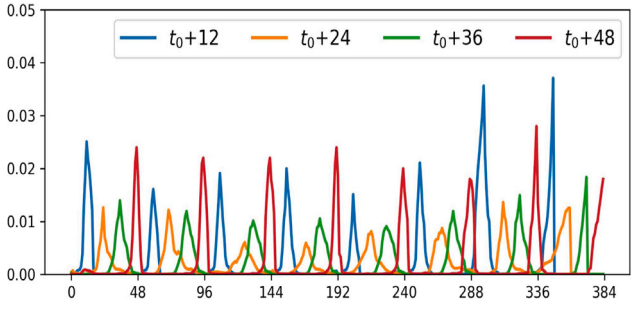



(b) Prediction Window

**Fig. 5.** The global feature importance learned by the proposed InterFormer.




(a) The first layer




(b) The last layer

**Fig. 6.** The temporal patterns learned by the proposed InterFormer.

from raw training data via an interpretable self-attention layer. Fig. 6 shows the attention weight patterns of the first and last layers of the prediction model. We visualize the average attention weights on the training set for different prediction time steps (i.e., $t \in \{t_0 + 12, t_0 + 24, t_0 + 36, t_0 + 48\}$). In the first layer, we observe that the attention weights are generated only during the prediction window, and the closer to the prediction time step, the greater the weight. This means that the model first extracts information for the current window at the prediction time step. In the last layer, the model focuses on all inputs including condition windows and prediction windows. Attention weights exhibit a seasonal pattern, with attention peaks appearing at substantially equal intervals across different days. InterFormer learns these temporal dependencies directly from the input data without any artificial assumptions. This allows us to easily see where in the input sequence the model focuses attention at each layer at each prediction time step. This ability is very helpful in improving the credibility of the model. Alternatively, we can use these insights to check whether the model captures the true causal relationship between inputs and outputs.

## 5. Conclusions

In this paper, a multi-step probabilistic forecasting method based on Transformer model for short-term residential net load is studied. Electric loads at individual household exhibit higher volatility than loads at the aggregate level. The connection of PV power generation behind the meter exacerbates the uncertainty of load profile. We quantify this uncertainty using quantile forecasts to obtain forecast intervals for future net load. In addition, in order to improve the interpretability of the deep learning model while building an accurate residential net load forecasting model, we design a local variable selection network to extract effective information from different features of the original input. An interpretable sparse self-attention mechanism is proposed to capture temporal dependencies in input data. Experimental results show that the proposed model can effectively improve the prediction performance. In terms of point forecasting, compared with LSTM, the proposed model decreases by 21.3%, 27.3% and 20.9% in terms of RMSE, MAE and MAAPE, respectively. Compared with Vanilla Transformer, the proposed InterFormer decreases by 9.9%, 10.5% and 9.0% in terms of RMSE, MAE and MAAPE, respectively. In terms of probabilistic forecasting, compared with LSTM and Vanilla Transformer, the average pinball loss of the proposed InterFormer decreases by 26.2% and 17.0%, respectively. In addition, the proposed InterFormer provides interpretability in terms of (i) important features for prediction, and (ii) visualizing temporal patterns learned by different layers through attention. These insights can help end users better understand the logic of the model, which is of great significance for the practical application of residential net load forecasting model.

## CRediT authorship contribution statement

**Chongchong Xu:** Investigation, Modeling, Simulation, Writing – original draft. **Guo Chen:** Conceptualization , Methodology, Supervision, Validation, Writing – review & editing.

## Declaration of competing interest

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.

## Data availability

The data that has been used is confidential.

## References


[1] Collier SE. The emerging enernet: Convergence of the smart grid with the internet of things. IEEE Ind Appl Mag 2016;23(2):12–6.
[2] Li C, Dong Z, Ding L, Petersen H, Qiu Z, Chen G, Prasad D. Interpretable memristive LSTM network design for probabilistic residential load forecasting. IEEE Trans Circuits Syst I Regul Pap 2022;69(6):2297–310.
[3] Ji Y, Buechler E, Rajagopal R. Data-driven load modeling and forecasting of residential appliances. IEEE Trans Smart Grid 2019;11(3):2652–61.
[4] Liu N, Yu X, Wang C, Li C, Ma L, Lei J. Energy-sharing model with price-based demand response for microgrids of peer-to-peer prosumers. IEEE Trans Power Syst 2017;32(5):3569–83.
[5] Ratnam EL, Weller SR, Kellett CM, Murray AT. Residential load and rooftop PV generation: an Australian distribution network dataset. Int J Sustain Energy 2017;36(8):787–806.
[6] Kong W, Dong ZY, Hill DJ, Luo F, Xu Y. Short-term residential load forecasting based on resident behaviour learning. IEEE Trans Power Syst 2017;33(1):1087–8.
[7] Sun M, Zhang T, Wang Y, Strbac G, Kang C. Using Bayesian deep learning to capture uncertainty for residential net load forecasting. IEEE Trans Power Syst 2019;35(1):188–201.
[8] Hong T, Fan S. Probabilistic electric load forecasting: A tutorial review. Int J Forecast 2016;32(3):914–38.
[9] Alavi SA, Ahmadian A, Aliakbar-Golkar M. Optimal probabilistic energy management in a typical micro-grid based-on robust optimization and point estimate method. Energy Convers Manage 2015;95:314–25.
[10] Lee D, Shin H, Baldick R. Bivariate probabilistic wind power and real-time price forecasting and their applications to wind power bidding strategy development. IEEE Trans Power Syst 2018;33(6):6087–97.
[11] Toubeau J-F, Bottieau J, Vallée F, De Grève Z. Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets. IEEE Trans Power Syst 2018;34(2):1203–15.
[12] Patsakos I, Vrochidou E, Papakostas GA. A survey on deep learning for building load forecasting. Math Probl Eng 2022;2022.
[13] Razavi SE, Arefi A, Ledwich G, Nourbakhsh G, Smith DB, Minakshi M. From load to net energy forecasting: Short-term residential forecasting for the blend of load and PV behind the meter. IEEE Access 2020;8:224343–53.
[14] Sehovac L, Grolinger K. Deep learning for load forecasting: Sequence to sequence recurrent neural networks with attention. IEEE Access 2020;8:36411–26.
[15] Imani M. Electrical load-temperature CNN for residential load forecasting. Energy 2021;227:120480.
[16] Huang J, Pang C, Yang W, Zeng X, Zhang J, Huang C. A deep learning neural network for the residential energy consumption prediction. IEEJ Trans Electr Electron Eng 2022;17(4):575–82.
[17] Wu S, Xiao X, Ding Q, Zhao P, Wei Y, Huang J. Adversarial sparse transformer for time series forecasting. Adv Neural Inf Process Syst 2020;33:17105–15.
[18] Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 35. 2021, p. 11106–15.
[19] L'Heureux A, Grolinger K, Capretz MA. Transformer-based model for electrical load forecasting. Energies 2022;15(14):4993.
[20] Wang C, Wang Y, Ding Z, Zheng T, Hu J, Zhang K. A transformer-based method of multienergy load forecasting in integrated energy system. IEEE Trans Smart Grid 2022;13(4):2703–14.
[21] Ran P, Dong K, Liu X, Wang J. Short-term load forecasting based on CEEMDAN and transformer. Electr Power Syst Res 2023;214:108885.
[22] Xu C, Li C, Zhou X. Interpretable LSTM based on mixture attention mechanism for multi-step residential load forecasting. Electronics 2022;11(14):2189.
[23] Koenker R. Quantile regression: 40 years on. Annu Rev Econom 2017;9:155–76.
[24] Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang Y-X, Yan X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In: Proceedings of the 33rd international conference on neural information processing systems. 2019, p. 5243–53.
[25] Chen Y, Kang Y, Chen Y, Wang Z. Probabilistic forecasting with temporal convolutional neural network. Neurocomputing 2020;399:491–501.
[26] Toubeau J-F, Bottieau J, Wang Y, Vallée F. Interpretable probabilistic forecasting of imbalances in renewable-dominated electricity systems. IEEE Trans Sustain Energy 2021;13(2):1267–77.
[27] Lin Y, Koprinska I, Rana M. Temporal convolutional attention neural networks for time series forecasting. In: 2021 international joint conference on neural networks (IJCNN). IEEE; 2021, p. 1–8.
[28] Peters B, Niculae V, Martins AF. Sparse sequence-to-sequence models. In: Proceedings of the 57th annual meeting of the association for computational linguistics. 2019, p. 1504–19.
[29] Martins AFT, Astudillo RF. From softmax to sparsemax: A sparse model of attention and multi-label classification. In: Proceedings of the 33rd international conference on international conference on machine learning - Vol. 48. JMLR.org; 2016, p. 1614–23.
[30] Tang X, Chen H, Xiang W, Yang J, Zou M. Short-term load forecasting using channel and temporal attention based temporal convolutional network. Electr Power Syst Res 2022;205:107761.
[31] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In: Proceedings of the 3rd international conference on learning representations (ICLR). 2015.
[32] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Adv Neural Inf Process Syst 2017;30.
[33] Treviso M, Góis A, Fernandes P, Fonseca E, Martins AF. Predicting attention sparsity in transformers. In: Proceedings of the sixth workshop on structured prediction for NLP. 2022, p. 67–81.
[34] Lim B, Arık SÖ, Loeff N, Pfister T. Temporal fusion transformers for interpretable multi-horizon time series forecasting. Int J Forecast 2021;37(4):1748–64.
[35] Fan C, Wang J, Gang W, Li S. Assessment of deep recurrent neural network-based strategies for short-term building energy predictions. Appl Energy 2019;236:700–10.
[36] Li A, Xiao F, Zhang C, Fan C. Attention-based interpretable neural network for building cooling load prediction. Appl Energy 2021;299:117238.
[37] Lamb AM, Goyal A, Zhang Y, Zhang S, Courville AC, Bengio Y. Professor forcing: A new algorithm for training recurrent networks. Adv Neural Inf Process Syst 2016;29.
[38] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. Pytorch: An imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 2019;32:8026–37.
[39] Winkler RL. A decision-theoretic approach to interval estimation. J Amer Statist Assoc 1972;67(337):187–91.