A Novel Sequence-to-Sequence based Deep Learning Model for Multi-step Load Forecasting

Renzhi Lu, *Member, IEEE,* Ruichang Bai, Ruidong Li, *Senior Member, IEEE,* Lijun Zhu, Mingyang Sun, *Senior Member, IEEE,* Feng Xiao, *Member, IEEE,* Dong Wang, *Senior Member, IEEE,* Huaming Wu, *Senior Member, IEEE,* Yuemin Ding, *Senior Member, IEEE*

***Abstract*—Load forecasting is critical to the task of energy management in power systems, for example, balancing supply and demand and minimizing energy transaction costs, etc. There are many approaches used for load forecasting such as the support vector regression, the autoregressive integrated moving average and neural networks, but most of these methods focus on single-step load forecasting, whereas, multi-step load forecasting can provide better insights for optimizing the energy resource allocation and assisting the decision-making process. In this work, a novel sequence-to-sequence based deep learning model based on a time series decomposition strategy for multi-step load forecasting is proposed. The model consists of a series of basic blocks, each of which includes one encoder and two decoders; and all basic blocks are connected by residuals. In the inner of each basic block, the encoder is realized by temporal convolution network for its benefit of parallel computing, and the decoder is implemented by long short-term memory neural network to predict and estimate time series. During the forecasting process, each basic block is forecasted individually. The final forecasted result is the aggregation of the predicted results in all basic blocks. Several cases within multiple real-world datasets are conducted to evaluate the performance of the proposed model. The results demonstrate that the proposed model achieves the best accuracy compared with several benchmark models.**

***Index Terms*—Multi-step load forecasting, sequence to sequence model, decomposition strategy, long short-term memory neural network, temporal convolution network.**

This work was supported in part by the National Natural Science Foundation of China under Grant 62003143, Grant 62373158, in part by the Natural Science Foundation of Hubei Province under Grant 2022CFB041, in part by the Wuhan Science and Technology innovation special project under Grant 2022010801020099, in part by the Fundamental Research Funds for the Central Universities under Grant HUST 2020kfyXJJS084, Grant 2022SMECP03, in part by the Key Laboratory of System Control and Information Processing under Grant Scip202211, in part by the Key Laboratory of Industrial Internet of Things and Networked Control under Grant 2020FF02, in part by the Open Fund of Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering at Wuhan University of Science and Technology under Grant MTMEOF2021B04, in part by the Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems under Grant ACIA2022001, and in part by the 111 project under Grant B17040. (Corresponding authors: Huaming Wu, Yuemin Ding.)

Renzhi Lu is with the Key Laboratory of Image Processing and Intelligent Control, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China, with the Key Laboratory of System Control and Information Processing, Ministry of Education, Shanghai 200240, China, with the Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China, and with the Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan 430081, China (e-mail: rzlu@hust.edu.cn).

Ruichang Bai is with the Central Academe, Shanghai Electric Group Co., Ltd, Shanghai 200070, China (email: bairch@shanghai-electric.com).

Ruidong Li is with the Institute of Science and Engineering, Kanazawa University, Kakuma, Kanazawa 920-1192, Japan (e-mail: liruidong@ieee.org).

Lijun Zhu is with the School of Artificial Intelligence and Automation, State Key Laboratory of Intelligent Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: ljzhu@hust.edu.cn).

Mingyang Sun is with the State Key Laboratory of Industrial Control Technology, Department of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: mingyangsun@zju.edu.cn).

Feng Xiao is with the State Key Laboratory of Alternate Electrical Power System with Renewable Energy Sources and the School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China (e-mail: fengxiao@ncepu.edu.cn).

Dong Wang is with the Key Laboratory of Intelligent Control and Optimization for Industrial Equipment of Ministry of Education, Dalian University of Technology, Dalian 116024, China, and also with the School of Control Science and Engineering, Dalian University of Technology, Dalian 116024, China (e-mail: dwang@dlut.edu.cn).

Huaming Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

Yuemin Ding is with the Department of Electrical and Electronic Engineering, University of Navarra, San Sebastian 20018, Spain (e-mail: yuemin.ding1986@gmail.com).

I. Introduction

With the rapid development of modernization and urbanization, the role of a stable and efficient power system in society has become increasingly significant [1]. In power systems, load forecasting is performed to determine the supply of electricity, which is essential to establish an efficient and accurate operating plan to reduce system losses and improve the reliability, efficiency and security of the electricity supply for customers [2]. Specifically, for energy suppliers, forecasting electricity loads in advance is critical to balancing production and demand, reducing production costs, and implementing various demand response pricing schemes [3]. Recently, an interesting and fantastic IEEE dataport day-ahead electricity demand forecasting competition was held [4], which was based on the post-COVID paradigm load data. The competition aimed to encourage the development and promotion of state-of-the-art load forecasting methods that can alleviate the negative impact of pandemic-related demand uncertainties on the electricity market. However, due to various uncertainties and the complexity of climate change, industrial structures and other social environments, load data exhibit complex patterns and fluctuations, making accurate load forecasting a challenging task [5].

Generally, according to forecasting horizon, the load forecasting can be classified into two categories: single-step ahead and multi-step ahead [6]. The single-step ahead load forecasting is use historical load data and related variables to predict the next one step load data, and the multi-step ahead forecasting is to predict future multi-step load data. Compared to single-step ahead load forecasting, the multi-step ahead load forecasting task presents various additional challenges, such as increased prediction errors and reduced accuracy, making the task of multi-step ahead forecasting more difficult [7]. Load data are essentially time-series. Many time series forecasting approaches have been reported in the literature. Broadly, these techniques can be grouped into three major groups: statistical methods, traditional machine learning-based methods, and deep learning-based methods [8].

Statistical models are based on fitting a regression model to the previous data and then validating the model by finding the difference between the actual and predicted values [9]. In particular, the autoregressive integrated moving average (ARIMA) model is one of the most famous statistical models used to perform load forecasting [10], and the ARIMA model overcomes the drawback that the autoregressive moving average (ARMA) model is only applicable to stationary time series [11]. Besides, in order to incorporate exogenous variables into the forecasting model, the autoregressive integrated moving average exogenous model (ARIMAX) has been proposed. The ARIMA and its variant models are simple and interpretable, making them easier to implement and understand, and could also achieve a good forecasting performance on the linear data, to a certain extent. However, a drawback is thereupon occurred for their assumption of a linear correlation structure in the underlying time series data. This proves unreasonable in many real-world load forecasting problems, since real-world data is often composed by both linear and nonlinear patterns [12].

Traditional machine learning-based models learn patterns from input load data. Different from statistical models, the input-output mapping in machine learning models does not need to be defined in advance. Instead, it is learned during the training process [13]. The support vector regression (SVR), which maps historical load data to a higher dimensional space through a nonlinear mapping and then performs linear regression on the mapped elements, is one of the most commonly used machine learning models for load forecasting [14]. Another commonly adopted model is the regression trees (RT) [15], which has a tree-like structure and regresses decisions in the form of a tree, starting from the root node down to the leaf nodes, where the leaf nodes contain the responses. Although the parameters of the SVR and RT are learned during the training process, feature extraction is still needed to perform to determine the inputs to the model. Moreover, adjusting the hyper-parameters of machine learning-based models would have a significant impact on its learning speed and performance, depending on the characteristics of the training data [16]. However, there is currently no standard method for selecting the best values for these hyper-parameters [17]. Typically, practitioners tune a subset of these parameters through trials to maximize accuracy on a validation dataset. In the specific study of this work, the hyper-parameters of the proposed model are given in Subsection D of Section V, which are established according to some empirical guidelines in the literature and a number of preliminary accuracy tests via trial and error. In addition, finding a well-fit training dataset is important for building a successful machine learning model. Including too much or too little information in the training data can have a crucial impact on the prediction accuracy [18]. If too few features are considered, the model will be simple, leading to high bias and low variance, this is known as underfitting. Underfitting is not fitting accurately in the data set via simple curve and linear hypothesis thus should always be low biased to avoid the problem of underfitting. On the contrary, if too many features are included, the real valuable features may be overshadowed by disturbances, resulting in a complex model, leading to high variance and low bias, this is known as overfitting. Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high [19]. Both underfitting and overfitting can decrease prediction accuracy. Since the quantity and quality of input features play a crucial role in forecasting accuracy. Various approaches, such as correlation analysis and principal components analysis, have been used to extract and select input features [20].

Deep learning-based models are promising approaches for accurate load forecasting due to their excellent nonlinear approximation capabilities enabling them to extract features well and automatically build complex mapping relationships between multiple inputs and outputs [2]. The convolutional neural network (CNN) and long short-term memory (LSTM) have been widely used in load prediction and have achieved some success in the field of single-step load forecasting. In addition, some hybrid methods combining CNN and LSTM for load forecasting have been proposed to take full advantage of the respective strengths of the CNN and LSTM [21]. In general, when proposing a deep learning-based forecasting model, it often require a large amount of data to train. Fortunately, the widespread deployment of smart meters in power grids has resulted in the availability of large amounts of data. Therefore, neural network models are currently recognized as one of the most promising approaches for power load forecasting.

The aforementioned traditional machine learning and deep learning-based models have achieved undeniable results in load forecasting, but they all focus on single-step ahead load forecasting. In most real-world applications, multi-step ahead forecasting is more valued than single-step ahead forecasting since it can provide key insights for optimizing the energy resource allocation and assisting the decision-making process [22].

Currently, there are several strategies for generating multi-step ahead forecasts: the Direct strategy, Recursive strategy and DirRec strategy [23]. However, these strategies are limited by their inherent flaws, and none of them can achieve good performance. Fortunately, the Multi-Input Multi-Output (MIMO) strategy implemented by sequence-to-sequence (Seq2Seq) model is regarded as a promising approach for multi-step ahead forecasting [24]. The heart of this model lies in two different sequential-based neural networks, namely, encoder and decoder, which can enhance the prediction of continuous sequences while also allowing the input and output to have different time dimensions. The encoder is responsible for converting the input sequence into a fixed-size vector representation, called the context vector. The decoder is responsible for converting the context vector into the output sequence. However, for longer input sequences, the encoder may suffer from incomplete compression, and it is difficult for the decoder to extract all the valuable information from the context vector.

To overcome these drawbacks of multi-step ahead forecasting models, a novel Seq2Seq-based deep learning model is proposed in this work. The model consists of a series of basic blocks, each of which is responsible for predicting a portion of patterns in the time series; and the basic blocks are connected by residuals. The residual removes the patterns that can be fitted well in the previous basic block, allowing the downstream basic block can concentrate on predicting patterns that are not learned by the previous basic block. The final prediction result is the aggregation of all basic blocks. To verify the effectiveness and evaluate the accuracy of the proposed model, multiple cases are conducted on real-word datasets. The results demonstrated that the proposed model outperforms all benchmark models in terms of accuracy. To better clearly clarify the advantages and drawbacks of the proposed model with other existing model, a comparative analysis is carried out with respect to some aspects as given in Table I, including application scene and performance comparison. In summary, this work has the following contributions:

1) A novel Seq2Seq-based deep learning model is proposed for multi-step ahead load forecasting, in which each basic block of the model is connected by residuals, and the final residual output is used as part of the loss function.
2) A decomposition strategy is deeply integrated into the Seq2Seq framework to improve the trainability of the deep architecture and the convergence of the model.
3) The proposed model dynamically decomposes the original time series into individual components for prediction, reducing the overall prediction burden and improving the forecasting accuracy.
4) In each basic block, temporal convolutional network (TCN) is used as the encoder, and LSTM is used as the decoder. The advantages of the TCN and LSTM are combined, and a considerable improvement in the results is achieved.

The rest of the paper is organized as follows: Section II presents the problem formulation for the load forecasting task, which consists of problem description and existing available approaches. Section III presents the technical preliminaries, which are the basis of the proposed model. Section IV describes the proposed methodology in detail. Section V explains the experiments and analyzes the corresponding results; and finally, Section VI concludes the paper.

II. Problem Formulation

In this section, we present the load forecasting task in detail in subsection A and analyze the existing multi-step ahead forecasting approaches in subsection B.

A. Problem Description

Assume that the time-series load data are $\boldsymbol{X}$, which can be described as follows:

$$\boldsymbol{X} = \{x_1, x_2, \ldots, x_T\} \tag{1}$$

where x_i represents the actual load value at the i-th timestamp, and T denotes the length of the historical load sequence. Load forecasting is to predict load value for the next H time steps given historical load data of length T, and the parameter H is called the forecasting horizon. When $H = 1$, it is called single-step ahead forecasting; and when $H > 1$, it is called multi-step ahead forecasting. More specifically, for single-step ahead forecasting, the formula can be expressed as follows:

$$x_{T+1} = F_{\text{single}}(x_1, x_2, \ldots, x_T) \tag{2}$$

where F_{single} denotes the single-step ahead forecasting model and x_{T+1} is the predicted load value given the historical load sequence. Different from the single-step ahead forecasting, the multi-step ahead forecasting is to predict various-step load values, such as k-step ahead forecasting as follows:

$$x_{T+1}, x_{T+2}, \ldots, x_{T+k} = F_{\text{multi}}(x_1, x_2, \ldots, x_T) \tag{3}$$

where F_{multi} denotes the multi-step ahead forecasting model. $x_{T+1:T+k}$ denotes the predicted sequence, including k predicted load values.

B. Existing Available Approaches

Currently, to the best of our knowledge, there are four main strategies that can perform multi-step ahead forecasting, i.e., the Recursive strategy, the Direct strategy, the DirRec strategy and the MIMO strategy [23]. Among the four strategies, the first three are called single-output models because they all establish a multiple-input single-output mapping. The last MIMO strategy is usually implemented by the Seq2Seq architecture. These four strategies are described and discussed as follows.

The Recursive strategy is the most intuitive and traditional forecasting strategy. It uses a single-step ahead forecasting model (such as SVR, ARIMA, or deep neural network) that recursively takes previous predictions as its input to predict subsequent predictions until H predictions are output [25]. The formula of the Recursive strategy for multi-step ahead forecasting is defined as follows:

$$\begin{cases} y_{t+1} = F_{\text{Rec}}\left(x_t, x_{t-1}, \ldots, x_{t-N+1}\right) \\ y_{t+2} = F_{\text{Rec}}\left(y_{t+1}, x_t, x_{t-1}, \ldots, x_{t-N+2}\right) \\ \vdots \\ y_{t+H} = F_{\text{Rec}}\left(y_{t+H-1}, y_{t+H-2}, \ldots, y_{t+H-N}\right) \end{cases} \tag{4}$$

where F_{Rec} denotes a single-step ahead forecasting model in the Recursive strategy. $y_{t+1:H}$ denotes the forecasted value, and $x_{t:t-N+1}$ denotes the given historical time series data of length N. Although the Recursive strategy is easy to implement, iteration-based forecasting methods will produce large cumulative errors when the forecasting horizon H is large [26].

Different from the Recursive strategy, the Direct strategy builds H different single-step ahead forecasting models for each forecasting horizon, and the Direct strategy is defined by the following equation:

$$y_{t+h} = F_{\text{Dir-h}}(x_t, x_{t-1}, \ldots, x_{t-N+1}) \tag{5}$$

TABLE I
THE COMPARISON BETWEEN THE PROPOSED APPROACH WITH OTHER EXISTING MODELS

Forecasting Models	Input Variables	Number of Parameters	Multi-Step Forecasting	Advantages	Disadvantages
Statistical Models	Historical time series data	2-3 (p, d, q)	Yes	Easy to use and highly interpretable	Limited to linear relationships, not suitable for complex relationships, and may be affected by outliers
Machine Learning Models	Historical time series data, other features	Many (hundreds to thousands)	No	Can handle various non-linear relationships and use external features	Requires extensive parameter tuning and feature engineering
LSTM, CNN, TCN, …	Historical time series data, other features	Many (tens of thousands to millions)	No	Can handle various non-linear relationships	Requires a significant amount of training data, training time, and computational resources.
Seq2Seq	Historical time series data, other features	Many (tens of thousands to millions)	Yes	Suitable for sequence-to-sequence forecasting tasks	Lack long-term memory capacity and gradually forget previous information
Proposed	Historical time series data, other features	Many (tens of thousands to millions)	Yes	High accuracy with time series additive decomposition	High computational complexity and long processing time

where $h \in \{1, 2, 3, \dots, H\}$, and $F_{\text{Dir-h}}$ denotes the h-th forecasting model. Obviously, the Direct strategy does not accumulate errors. However, since H models are learned independently, which prevents this approach from considering the complex dependencies between the predicted values, and this further affects the forecasting accuracy [26].

The DirRec strategy combines the two previous strategies, which use different single-step ahead forecasting models for each horizon to calculate the forecasts (similar in this respect to the Direct strategy), then, at each time step, expand the input set by adding variables corresponding to the forecasts of the previous step (similar in this respect to the Recursive strategy). The formula of this strategy is as follows:

$$\begin{cases} y_{t+1} = F_{\text{DirRec-1}}\left(x_t, x_{t-1}, \dots, x_{t-N+1}\right) \\ y_{t+2} = F_{\text{DirRec-2}}\left(y_{t+1}, x_t, x_{t-1}, \dots, x_{t-N+1}\right) \\ \vdots \\ y_{t+H} = F_{\text{DirRec-H}}\left(y_{t+H-1}, \dots, y_{t+1}, x_t, \dots, x_{t-N+1}\right) \end{cases} \tag{6}$$

where $F_{\text{DirRec-h}}$ denotes the h-th forecasting model and y_{t+h} denotes the forecasted value at moment $t + h$.

Although the DirRec strategy avoids the disadvantage of the Direct strategy, it still has the disadvantage of the Recursive strategy, i.e., error accumulation [25].

The MIMO strategy can address the inherent drawbacks of single-output strategies since the MIMO strategy models multiple-output dependencies rather than modeling single output mapping [27]. Consequently, the MIMO strategy can avoid error accumulation and overcome the problem caused by using independently single-step predictor at different steps in the Direct strategy. However, The MIMO strategy imposes a constraint that all horizons being predicted must use the same model structure and the same set of input training data. This limitation reduces the flexibility of the forecasting approach and may bias the resulting model [28]. Fortunately, the Seq2Seq framework is employed to implement the MIMO strategy, and the advantages of neural networks can significantly alleviate the drawbacks of the MIMO approach. The Seq2Seq framework is comprised of two sequential-based neural networks, an encoder and a decoder. However, for longer input sequences in Seq2Seq model, the encoder may suffer from incomplete compression and it is difficult for the decoder to extract all the valuable information from this single vector, which therefore affects the prediction accuracy.

To overcome the drawbacks of current multi-step ahead forecasting methodologies, a novel Seq2Seq-based deep learning model is proposed in this work, and the details are presented in section IV.

III. Technical Preliminaries

This section will briefly review the related basic deep learning techniques for time series forecasting, including LSTM, TCN and Seq2Seq. All of these techniques are preliminary knowledge for the proposed model.

A. LSTM

Benefiting from the self-feedback mechanism, the recurrent neural network (RNN) model has advantages in exploring the temporal relationships in time series. However, it is more prone to gradient disappearance in practical applications. LSTM is designed to solve this problem on the basis of RNN [29], and thus with the ability to build long term dependencies.

Fig. 1 shows the structure of a LSTM unit. In order to establish long-term dependence, LSTM maintains a new internal state $\boldsymbol{s}$ throughout its life cycle. In addition, three gate structures are introduced: input gate, forget gate, and output gate. The internal state $\boldsymbol{s}_{t-1}$ interacts with the external state $\boldsymbol{h}_{t-1}$ and the input $\boldsymbol{x}_t$. With the help of the gate mechanism, the output of the previous time step and the input of the current time step are used to determine the internal state vector of the elements that should be maintained, updated or deleted [30].

The update formula of the state in LSTM at t-th timestamp is as follows:

$$\begin{aligned}
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{fx}\boldsymbol{x}_t + \boldsymbol{W}_{fh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{ix}\boldsymbol{x}_t + \boldsymbol{W}_{ih}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \\
\boldsymbol{g}_t &= \phi(\boldsymbol{W}_{gx}\boldsymbol{x}_t + \boldsymbol{W}_{gh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_g) \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{ox}\boldsymbol{x}_t + \boldsymbol{W}_{oh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \\
\boldsymbol{s}_t &= \boldsymbol{g}_t \odot \boldsymbol{i}_t + \boldsymbol{s}_{t-1} \odot \boldsymbol{f}_t \\
\boldsymbol{h}_t &= \phi(\boldsymbol{s}_t) \odot \boldsymbol{o}_t
\end{aligned} \tag{7}$$

where $\boldsymbol{W}_{fx}$, $\boldsymbol{W}_{fh}$, $\boldsymbol{W}_{ix}$, $\boldsymbol{W}_{ih}$, $\boldsymbol{W}_{gx}$, $\boldsymbol{W}_{gh}$, $\boldsymbol{W}_{ox}$ and $\boldsymbol{W}_{oh}$ are the weight matrices; and $\boldsymbol{b}_f$, $\boldsymbol{b}_i$, $\boldsymbol{b}_g$ and $\boldsymbol{b}_o$ are the biases of the activation function. $\sigma(\cdot)$ is the sigmoid activation function, $\phi(\cdot)$ represents the hyperbolic tangent function, and $\odot$ represents the element-by-element multiplication. $\boldsymbol{i}_t$, $\boldsymbol{f}_t$ and $\boldsymbol{o}_t$ denote the states of input gate, forget gate and output gate, respectively; and $\boldsymbol{g}_t$ is the candidate state [31].

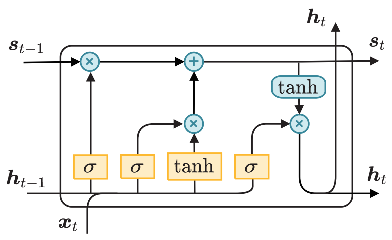


Fig. 1. The structure of LSTM unit.

B. TCN

The TCN is a general convolution model for sequence modeling tasks with powerful feature extraction and efficient parallel computing capabilities [32]. It consists of three parts: causal convolution, dilated convolution and residual connection.

The causal convolutions ensure causal constraint, which means that there is no leakage of information from the future to past. Furthermore, to expand the receptive field of causal convolution more efficiently, the dilated convolution, which can realize an exponentially enlarged receptive field, is employed in the TCN [32]. Fig. 2a shows a dilated and causal convolution with a convolution kernel size of 2 and an exponential increase in expansion factor $d(d = O(2^i))$ in the i-th layer of the network.

In addition, in the design of the general TCN model, a residual block is adopted to replace the convolutional layer. Fig. 2b shows the residual block of the general TCN. To ensure that the addition of residual blocks can accept tensors of the same shape, a 1×1 convolution kernel is adopted at the residual connection to perform dimensional transformation. Within the residual block, the TCN has two dilation causal convolution layers (Dilated Causal Conv), activation functions (ReLU) and weight normalization. Furthermore, in order to avoid overfitting, the spatial dropout after each causal dilation convolution layer is employed for regularization [32].

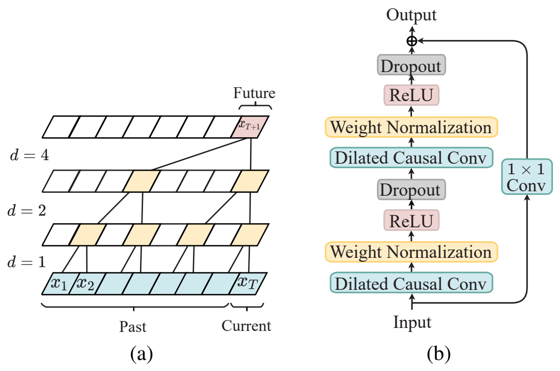


Fig. 2. (a) A dilated causal convolution; (b) TCN residual block.

C. Seq2Seq Model

The Seq2Seq structure was originally designed to solve the problem that the RNN cannot generate arbitrary length output sequences in neural machine translation. The core idea of the Seq2Seq model is using two networks to form an encoder-decoder architecture. The encoder is responsible for converting the input sequence into a fixed-size vector, called the context vector. The decoder is responsible for converting the context vector into the output sequence [33].

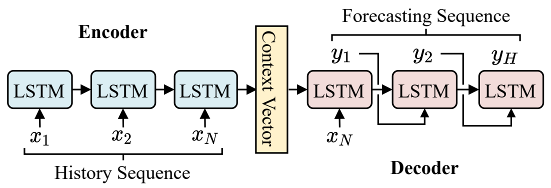


Fig. 3. The Seq2Seq model based LSTM encoder and decoder.

Codecs are typically multi-layer LSTM structures due to the natural convenience of LSTM to process sequence data. Fig. 3 shows the common Seq2Seq forecasting model. The codecs are all composed of a chain of LSTM units. In the encoder, the input is historical sequence data $\boldsymbol{X} = \{x_1, x_2, x_3, \ldots, x_N\}$, and the context vector is the output state of the last LSTM unit. For the decoder, the input of the first LSTM unit is x_N, which is the last time step input of the encoder; and each unit input thereafter is the predicted output value of the previous unit. The length of the input sequence of the encoder and the length of the output sequence of the decoder can be different, and the model will automatically learn the mapping relationship between the input sequence and the forecasting sequence.

However, in the original Seq2Seq structure, for longer input sequences, the encoder may not be able to encode all valuable timing features into this vector, and it is difficult for the decoder to extract all valuable timing features. To overcome this shortcoming, a novel Seq2Seq model, which is based on a time series additive decomposition strategy, is proposed in this work.

IV. Proposed Model

In this section, we describe the proposed model, which is based on the time series additive decomposition strategy and original Seq2Seq structure. First, the overall architecture

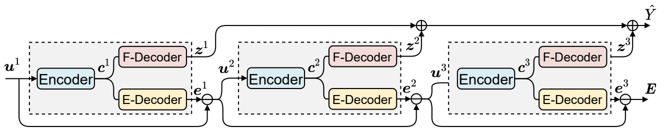


Fig. 4. The architecture of proposed model, consisted a series of basic blocks.

is described, and then the internal details of this model are presented.

A. Overall Architecture

In the task of time series forecasting, an effective approach is to decompose the time series into multiple components and then model each component individually [34]. Additive decomposition is a classical time series decomposition method, which assumes that time series data can be decomposed into seasonal, trend-cycle, and remainder components [35]. When applying the additive decomposition method to decompose the time series and subsequently model each component separately, it reduces the sensitivity of the prediction model to the noise in the time series and enhances the robustness of the prediction model [36]. Additionally, there are more robust and efficient decomposition approaches, such as Robuststl [37] and Fast Robuststl [38]. However, these explicit decompositions divide the time series into fixed components in advance and separate them from the modeling task, which fails to adequately address the temporal characteristics present in real-world data. Different from the native decomposition strategy, the model proposed in this work dynamically decomposes the time series into any number of components during the forecasting process.

Fig. 4 illustrates the overall proposed framework, consisting of a series of basic blocks (exemplified here by three basic blocks for brevity), and each of which can be viewed as a component of a decomposition. Each basic block includes three parts and appears as a fork-like structure. The first part is the encoder, which is responsible for compressing all the useful information of the input sequence data $\boldsymbol{u}$ into the context vector $\boldsymbol{c}$. The second part is the forecasting decoder (F-Decoder), which decodes the context vector $\boldsymbol{c}$ to produce the forecasting output $\boldsymbol{z}$. And the third part is the estimate decoder (E-Decoder), which is responsible for generating the best estimate $\boldsymbol{e}$ of the input sequence $\boldsymbol{u}$ based on the context vector $\boldsymbol{c}$, where the length of the estimate sequence $\boldsymbol{e}$ is the same as the length of the input sequence $\boldsymbol{u}$.

To have a clear understanding of the proposed model, Fig. 5 presents an example using actual time series data, demonstrating the functioning of each basic block during forecasting (exemplified here by three basic blocks for brevity). Specifically, for the i-th basic block, it receives a sequence data $\boldsymbol{u}^i \in R^{N*d}$ and outputs two sequence data, forecasting sequence $\boldsymbol{z}^i \in R^{H*1}$ and estimate sequence $\boldsymbol{e}^i \in R^{N*1}$. For the first basic block of the model, its input $\boldsymbol{u}^1$ is the original data $\boldsymbol{X} = \{x_1, x_2, x_3, \ldots, x_N\}$, which has N timestamps and each timestamp contains d features. For the other basic blocks, the inputs are the residuals of the previous basic block, which is obtained by subtracting its estimate $\boldsymbol{e}^{i-1}$ from the input sequence $\boldsymbol{u}^{i-1}$, therefore, each timestamp contains one feature, i.e., $d = 1$. The one of the outputs, $\boldsymbol{z}^i$, is the forecast sequence data of length H generated by the F-Decoder. And the another output, $\boldsymbol{e}^i$, is the estimated sequence of input data produced by the E-Decoder. Formally, the following equations are the computation process for the i-th ($i \geq 2$) block:

$$\begin{aligned} \boldsymbol{u}^i &= \boldsymbol{u}^{i-1} - \boldsymbol{e}^{i-1} \\ \boldsymbol{c}^i &= f(\boldsymbol{u}^i, \boldsymbol{\theta}_f^i) \\ \boldsymbol{z}^i &= g_f(\boldsymbol{c}^i, \boldsymbol{\theta}_{gf}^i) \\ \boldsymbol{e}^i &= g_e(\boldsymbol{c}^i, \boldsymbol{\theta}_{ge}^i) \end{aligned} \tag{8}$$

where $\boldsymbol{u}^i$ is the input of the i-th block; and $\boldsymbol{z}^i$ and $\boldsymbol{e}^i$, which are the forecast and estimate sequence data, respectively, are both the outputs of the i-th block. $f(\cdot)$, $g_f(\cdot)$ and $g_e(\cdot)$ represent the encoder, F-Decoder and E-Decoder respectively, and $\boldsymbol{\theta}_f^i$, $\boldsymbol{\theta}_{gf}^i$ and $\boldsymbol{\theta}_{ge}^i$ are the corresponding parameters.

Generally, the residual in a neural network refers to the difference or error between input and output of some layers or blocks in the model. The utilization of residual blocks can address the problem of gradient vanishing while propagating information from shallow to deep layers, and prevent performance degradation with deeper network structures [39]. Additionally, residual connections can enhance the accuracy and stability of the model [40]. In this work, however, in each basic block, the residual refers to the difference between the input and the estimated output. Then, the residual is regarded as the input of the next basic block, so that each subsequent basic block is only a prediction and analysis of the residuals of the previous basic block. The residual input removes the patterns from the sequence data that can be fitted well in the previous block, allowing the downstream basic block to concentrate more on predicting patterns that are not learned. Thus, the problems that the context vector in the original Seq2Seq model cannot fully express all the patterns of the input sequence and the difficulty for the decoder to extract all the useful information from the context vector are avoided. In addition, the basic blocks are connected by residuals, which has a significant advantage in improving the trainability of the deep architecture.

Correspondingly, the final output includes two items: one is $\hat{\boldsymbol{Y}}$, a prediction sequence of length H, which is the aggregation of all basic block forecast outputs; and the other is $\boldsymbol{E}$, a residual of length N, which will be used as part of the model loss value to ensure model convergence. The computational operation are as follows:

$$\begin{aligned} \hat{\boldsymbol{Y}} &= \sum_i^D \boldsymbol{z}^i \\ \boldsymbol{E} &= \boldsymbol{X} - \sum_i^D \boldsymbol{e}^i \end{aligned} \tag{9}$$

where $\boldsymbol{z}^i$ and $\boldsymbol{e}^i$ are the forecast and estimate output of the i-th block, respectively, and D is the total number of basic blocks.

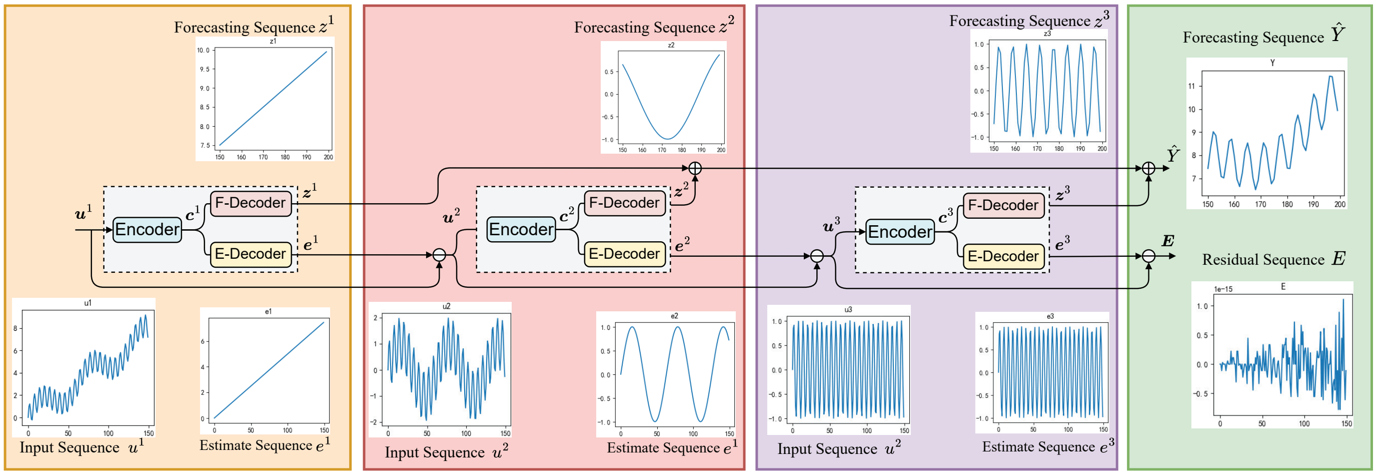


Fig. 5. The dynamic decomposition process of the proposed model.

B. Basic Block

As previous subsection mentioned, each basic block consists of three parts, i.e., one encoder and two decoders. TCN is adopted as the encoder due to its powerful feature extraction and efficient computation capabilities, which have been introduced in Section III. And the two decoders are a series of LSTM units with shared parameters, respectively. Fig. 6 shows the internal details of the basic block. In the encoder, the number of TCN residual blocks is a hyperparameter, which is determined by both the convolution field and the length of the input sequence. In addition, in the two decoders, the number of LSTM units is determined by the length of the forecasting sequence and estimate sequence, respectively.

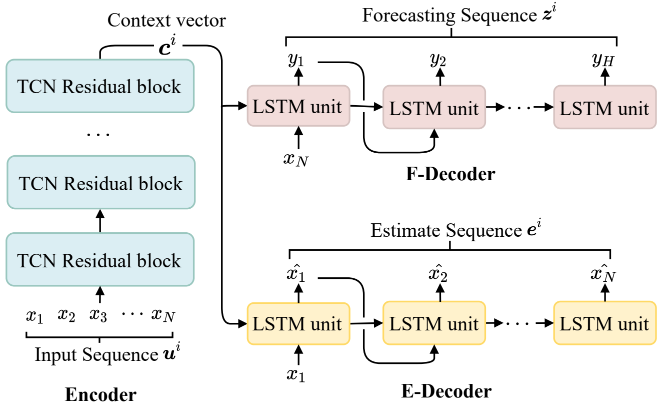


Fig. 6. The internal details of the i-th basic block.

For the i-th basic block, suppose the input sequence is $\boldsymbol{u}^i$, and the context vector $\boldsymbol{c}^i \in R^h$ is generated through the TCN encoder. Note that the context vector is only the output of the last time step of the last TCN residual block. Furthermore, the F-Encoder and E-Decoder share the context vector $\boldsymbol{c}^i$ to generate the predicted series data $\boldsymbol{z}^i$ and estimated series data $\boldsymbol{e}^i$, respectively. Specifically, for the F-Decoder, the input of the first LSTM unit is the last time step of the current basic block input sequence, and the input of the subsequent units is the forecast output of the previous unit. For the E-Decoder, since it is the estimate of the input sequence, the input of the first LSTM unit is the data of the first time step of the input to the current basic block.

C. Model Training

The model parameters are learned by back propagation mechanism, and the mean squared error (MSE) is selected as the loss function. In addition, the residual output of the model is added to the loss function, so that the model can learn the pattern contained in the input sequence as completely as possible. The loss function is defined as follows:

$$L(\theta) = \frac{1}{m}\sum_{i}^{m}[(\boldsymbol{Y_i} - \hat{\boldsymbol{Y_i}})^2 \; + \; \boldsymbol{E_i}^2] \tag{10}$$

where m is the number of samples; and $\boldsymbol{Y_i}$, $\hat{\boldsymbol{Y_i}}$ and $\boldsymbol{E_i}$ are the true vector, prediction vector and residual vector of the i-th sample, respectively. The training process is provided in Algorithm1.

V. Experiments and Discussion

In this section, the effectiveness of the proposed model on real-world datasets with three cases is verified. Case 1 presents the superiority of the proposed model. Case 2 demonstrates the effect of the decomposition strategy. Case 3 explores the effect of model hyperparameter changes on forecast performance. Case 4 conducts accuracy comparison of the proposed model using the competition post-COVID-19 load demand data. Case 5 examines the model's convergence speed, and comparing it with other neural network models. Case 6, the impact of sliding window size on prediction accuracy is thoroughly investigated. Case 7 delves into the diverse choices available for the encoder and decoder of the proposed model.

A. Datasets Description

The datasets used in the experiments are obtained from the Open Power System Data Platform (https://data.open-power-system-data.org/), including historical load data

Algorithm 1 The training process of the proposed model

```
1:  Prepare and process the dataset.
2:  Initialize the hyperparameters, i.e., the total epochs $M$, number of basic block $D$, learning rate $r$, and the parameters $\boldsymbol{\theta}$ in the model $\Phi(\cdot)$
3:  for $epoch = 1$ to $M$ do
4:      Sample input $\boldsymbol{X} \in R^{N*d}$ and target $\boldsymbol{Y} \in R^{H*1}$ from datasets randomly
5:      for $i = 1$ to $D$ do
6:          $\boldsymbol{c}^i = f(\boldsymbol{u}^i, \boldsymbol{\theta})$    # Encoding
7:          $\boldsymbol{z}^i = g_f(\boldsymbol{c}^i, \boldsymbol{\theta})$    # Forecast Decoding
8:          $\boldsymbol{e}^i = g_e(\boldsymbol{c}^i, \boldsymbol{\theta})$    # Estimate Decoding
9:          $\boldsymbol{u}^{i+1} = \boldsymbol{u}^i - \boldsymbol{e}^i$    # Calculating residuals
10:     end for
11:     Aggregate forecast sequence $\hat{\boldsymbol{Y}} = \sum_i^D \boldsymbol{z}^i$
12:     Compute $loss$ with Eq. (10)
13:     Back propagation of $loss$
14:     $\boldsymbol{\theta} \leftarrow \text{Adam}(\boldsymbol{\theta}, r)$
15: end for
16: return Trained model $\Phi(\cdot)$
```

and weather data with a sampling rate of 1 hour in several different countries. Specifically, we select the load data of Switzerland (CH) from 2011 to 2014, the load data of Germany (DE) from 2014 to 2017 and the load data of Australia (AT) from 2013 to 2016. Besides, we selected the temperature and wind speed as relevant variable to assist load forecasting. The selection of input variables are based correlation analysis and some empirical guidelines described in [41] [42]. Thus, the datasets contain four parts, namely, timestamps, historical load data, temperatures, and wind speeds. The timestamp records the sampling times, the load data are the variables to be predicted, and the temperature and wind speed are the relevant variables to assist load forecasting. The datasets for the experiments and source code are stored on GitHub (https://github.com/BaiRuic/Novel-Seq2Seq-Module).

When conducting the experiments, the ratio of training datasets, validating datasets, and testing datasets is roughly divided into 0.8:0.1:0.1. Fig. 7 illustrates the correspond data patterns of the three countries used in this work.

B. Data Preprocessing

In order to keep all the data in the same range, maximum-minimum normalization is applied to all the variables in the datasets and is formulated as follows:

$$x\prime(t) = \frac{x(t) - x_{\min}}{x_{\max} - x_{\min}} \tag{11}$$

where $x(t)$ and $x\prime(t)$ represent the raw value and the normalized value at timestamp t, respectively. $x_{\min}$ and $x_{\max}$ represent the minimum and maximum values in all time steps of the feature, respectively.

As defined in Section II, time series forecasting is the modeling of the relationship between a set of input variables and one or more output variables on a set of observed data. However, the original load data are a sequence of historical measurements at equal time intervals, which cannot be used to train the model directly; therefore, in this work, a sliding window approach is used to transform the dataset into input and target pairs for training the model.

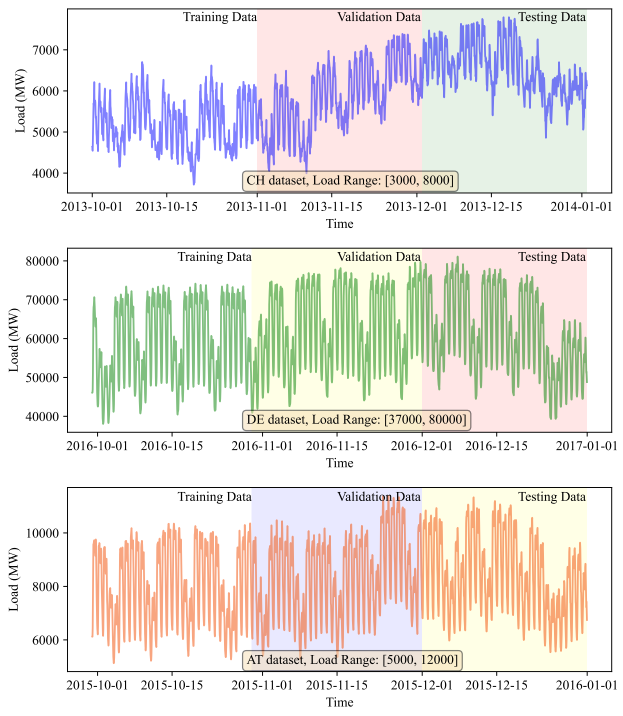
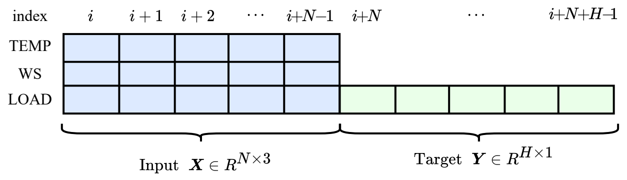


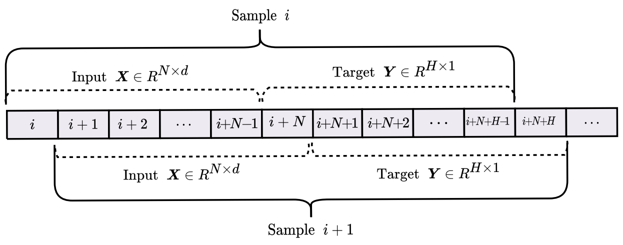
Fig. 7. The load data of three real-world datasets (CH, DE and AT).

Fig. 8 illustrates the input and target pairs in a sample of the multi-step ahead load forecasting task. The input sample can be represented as a matrix $\boldsymbol{X} \in R^{N\times d}$, where N denotes the number of input time steps and d denotes the number of features at each time step. In this work, the input data have three features, i.e., the load data (LOAD), temperature data (TEMP), and wind speed (WP). The target sample is a vector $\boldsymbol{Y} \in R^{H\times 1}$, where H denotes the forecast horizon. Fig. 9 shows the process of sample generation. When generating samples, a window of length $N + H$ slides over the time series, and each sliding of the window indicates that a sample is generated. In each generated sample, the vector of the first N time steps is the input, and the vector of the last H time steps is the target.


index
i
$i+1$
$i+2$
$\cdots$
$i{+}N{-}1$
$i{+}N$
$\cdots$
$i{+}N{+}H{-}1$
TEMP
WS
LOAD
Input $\boldsymbol{X} \in R^{N\times 3}$
Target $\boldsymbol{Y} \in R^{H\times 1}$
Fig. 8. The input and target pairs.

C. The Benchmarks

In order to better demonstrate the performance of the proposed model, some forecasting models are selected as benchmark models: the naive forecast (Naive), which is a



Sample i
Input $\boldsymbol{X} \in R^{N\times d}$
Target $\boldsymbol{Y} \in R^{H\times 1}$
i
$i+1$
$i+2$
$\cdots$
$i{+}N{-}1$
$i+N$
$i{+}N{+}1$
$i{+}N{+}2$
$\cdots$
$i{+}N{+}H{-}1$
$i{+}N{+}H$
$\cdots$
Input $\boldsymbol{X} \in R^{N\times d}$
Target $\boldsymbol{Y} \in R^{H\times 1}$
Sample $i+1$

Fig. 9. A sliding window approach is used to generate sample.

sample baseline used to measure the difficulty of forecasting task, statistical model, traditional machine learning model, and deep learning model.

Specifically, the ARIMA model, which was employed in [10] to predict loads, is selected as the statistical benchmark model. The ARIMA model contains three main components, autoregressive (AR), integrated (I) and moving average (MA). In the ARIMA model, the future variable is a linear function of past observations and some random errors, with the following equation:

$$y_t = \delta + \sum_{i=1}^{p} \phi_i y_{t-i} + a_t - \sum_{j=1}^{q} \theta_j a_{t-j} \tag{12}$$

where y_t and a_t are the true value and random error at timestamp t, respectively; ϕ and θ are the parameters of the model; and p and q are the orders of the model [10]. For this benchmark, the Recursive strategy is used to perform multi-step ahead forecasting.

The SVR is selected as the traditional machine learning benchmark model. By introducing kernel functions, the SVR can map the original feature space to a higher dimensional feature space, which converts the nonlinear problem in the original feature space to a linear problem in a high-dimensional feature space [43]. The SVR is formulated as follows:

$$f(x) = \boldsymbol{w}^{\mathrm{T}} \phi(x) + \boldsymbol{b} \tag{13}$$

where $f(\cdot)$ is the SVR model, $\phi(\cdot)$ is the kernel function, and $\boldsymbol{w}$ and $\boldsymbol{b}$ are the vector perpendicular to the separating hyperplane and the displacement of the separating hyperplane, respectively.

Besides, the MD-XGBoost, which is recently proposed by [44], is also selected as a benchmark model. An adaptive decomposition method based on Variational mode decomposition (VMD) and SampEn (SVMD) is adopted to firstly decompose the raw load data into a set of fluctuation sub-series. Then, the prediction model is correspondingly established for each fluctuation sub-series [44]. Similar to the ARIMA model, the Recursive strategy is used in SVR model and MD-XGBoost to perform multi-step prediction.

Finally, the original Seq2Seq model (LSTM-LSTM), which is introduced in Section III, is selected as the deep learning benchmark model. The selection of hyperparameter of the model is presented in the next subsection.

D. Implementation Details

For the proposed model and the benchmark models, some hyperparameters need to be predetermined before training. In this work, the grid search strategy is used to fine-tune the optimal values of the hyperparameters. Specifically, for each model, we first specify the hyperparameters to be set and the corresponding values to be tried. Then, the grid search strategy will test all possible combinations of hyperparameter values to build each model and determine the optimal combination of hyperparameters that yields the best performance. Table II lists the hyperparameters optimized for each model and the corresponding values.

To quantitatively assess the performance of the proposed model and benchmark methods, two common metrics: the mean squared error (MSE) and the mean absolute percentage error (MAPE), are chosen to evaluate forecasting performance:

$$\begin{aligned} \text{MSE} &= \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ \text{MAPE} &= \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \end{aligned} \tag{14}$$

where y represents the true value, $\hat{y}$ represents the forecast value, and n represents the number of samples.

All forecasting models are built on a desktop PC with a 3.4GHz Intel i5-7500 processor and 8GB of memory using the PyTorch library [45].

E. Case 1: Comparison With Benchmark Models

In this subsection, comparative experiments are conducted to verify the superiority of the proposed model. The proposed model and benchmark models are verified with four different forecasting horizons of $H = 3, 6, 12$ and 24 on three real-world datasets.

Table III shows the corresponding results, and it can be concluded that: First, for different forecasting horizons, the proposed model achieves the best performance in two evaluation metrics compared to benchmark models. For example, as shown in the second column of the CH dataset, the MAPE of the proposed model is 2.958%, while the benchmark models are $11.31\%, 3.439\%, 5.401\%, 4.532\%, 4.061\%$ when $H = 3$. Second, regarding the results on three different datasets, although the proposed model achieves the best performance, it is obvious that their metrics are much different. Specifically, on the CH dataset, the average MAPE of the proposed model is 4.38%; while on the DE dataset, the average MAPE is 7.23%; and on the AT dataset, the average MAPE is 5.47%. This phenomenon suggests that the prediction accuracy is related to the characteristics of the dataset itself. Furthermore, according to the result shown in Table III it can be observed that the MAPE of Naive model on the DE dataset is significantly bigger, for example, when $H = 3$, the MAPE of Naive model in three datasets are $11.31\%, 16.654\%, 9.891\%$. This phenomenon shows that accurate prediction on the DE dataset is more difficult than those on the AT and CH datasets. Besides, compared to the CH dataset, the DE dataset has relatively large nonstationary (see Fig. 7) and therefore has the worst average prediction results.

Moreover, as the forecasting horizon increases, the accuracy of all multi-step ahead forecasting models generally decreases.

TABLE II
THE HYPERPARAMETERS TO BE OPTIMIZED FOR EACH MODEL AND THE CORRESPONDING DESCRIBE AND VALUES

Model	Hyperparameters	Describes	Optional Values
ARIMA	p	The order of autoregression	{1, 2, 3, 4, 5}
	q	The order of moving average	{1, 2, 3, 4, 5}
	d	The degree of difference needed for stationarity	{1, 2}
SVR	kernel	The kernel function of SVR model	{'rbf', 'sigmoid', 'poly'}
	g	The coefficients of the kernel function	{0.05, 0.1, 0.5, 1, 5}
	d	Degree of the polynomial kernel function ('poly'). Ignored by other kernels.	{3, 5, 8}
	C	Regularization parameter	{1, 5, 10}
	e	Tolerance of termination criterion	{0.01, 0.1, 0.5, 1}
MD-XGBoost	eta	The step size / learning rate for each iteration	{0.08, 0.06, 0.05, 0.04}
	gamma	The minimum loss reduction required to make a further partition	{0.1, 0.4, 0.6, 1.2}
	max depth	The maximum depth of the tree	{4, 6, 8, 10}
	subsample	The proportion of random sampling for each tree	{0.3, 0.4, 0.5, 0.6, 0.7, 0.8}
	colsample bytree	The subsample ratio of columns when constructing each tree	{0.5, 0.6, 0.7, 0.8}
	min child weight	The minimum number of instances needed in each node	{0.8, 1.0, 1.2, 1.5}
LSTM-LSTM	hidden size	The number of features in the hidden state h	{6, 8, 12}
	activation function	The form of activation functions	{'tanh', 'Relu'}
	optimizer	The optimizer to minimize the loss function	{'Adam', 'SGD'}
Proposed	block size	The number of the basic block	{2, 4, 5, 6, 8, 10}
	residual block size	Size of the residual block in TCN encoder of each basic block	{3, 4, 6}
	kernel size	Size of the convolution kernel of TCN Encoder	{2, 3, 5}
	optimizer	The optimizer to minimize the loss function	{'Adam', 'SGD'}

TABLE III
PERFORMANCE COMPARISON WITH BENCHMARK MODELS

Prediction Horizon	Model	CH datasets		DE datasets		AT datasets	
		MSE($\cdot 10^{-2}$)	MAPE (%)	MSE($\cdot 10^{-2}$)	MAPE (%)	MSE($\cdot 10^{-2}$)	MAPE (%)
H = 3	Naive	9.200	11.31	11.60	16.654	10.80	9.891
	ARIMA	0.351	3.439	1.180	5.450	0.806	5.443
	SVR	0.194	5.401	0.794	6.812	0.679	5.681
	MD-XGBoost	0.186	4.532	0.352	5.411	0.252	3.198
	LSTM-LSTM	0.197	4.061	0.246	5.357	0.370	4.208
	Proposed	0.099	2.958	0.095	4.099	0.102	2.996
H = 6	Naive	11.90	14.33	21.60	22.128	16.20	15.644
	ARIMA	0.597	4.548	4.770	11.227	2.280	9.734
	SVR	0.610	10.998	2.030	11.304	0.829	7.012
	MD-XGBoost	0.520	5.659	1.790	7.623	0.601	6.018
	LSTM-LSTM	0.433	6.114	1.400	10.748	0.633	5.740
	Proposed	0.143	3.989	0.902	6.900	0.465	4.232
H = 12	Naive	19.80	20.62	26.10	32.048	19.90	21.762
	ARIMA	1.000	6.145	6.820	20.945	4.570	15.114
	SVR	1.480	14.255	3.210	14.890	0.967	7.287
	MD-XGBoost	0.973	10.189	2.120	11.152	1.010	7.831
	LSTM-LSTM	0.558	7.165	2.460	12.240	1.806	7.534
	Proposed	0.225	4.598	1.230	7.416	0.925	5.818
H = 24	Naive	18.20	20.120	25.20	32.018	20.20	22.653
	ARIMA	1.260	12.213	9.980	24.306	4.880	16.469
	SVR	2.560	19.009	3.540	16.129	3.560	15.937
	MD-XGBoost	1.940	11.156	2.980	13.114	2.120	12.129
	LSTM-LSTM	0.645	10.329	3.560	16.224	2.680	13.408
	Proposed	0.343	5.9820	2.150	10.524	1.500	8.826

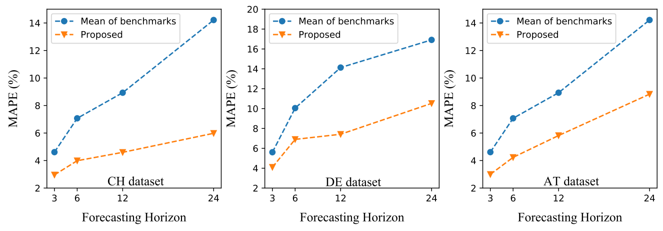


Fig. 10. Comparison of the mean of MAPE of all models with the MAPE of the proposed model in the four scenarios

Fig. 10 shows how the mean MAPE of benchmark models excluding the naive benchmark change as the forecasting horizon increases, and compared with the proposed model. The reason that accounts for this phenomenon mainly include two point, one is that as the forecasting horizon increases, the uncertainty of the load data increases, and the forecasting task becomes relatively more difficult. The second is due to the shortcomings of the forecasting model. Specifically, the SVR model and the ARIMA model adopt the Recursive strategy to perform multi-step ahead forecasting tasks, which has inherent error accumulation problems. Therefore, when the forecast horizon increases, the accuracy will drop sharply. Although the Seq2Seq model does not have the error accumulation problem, as the forecasting horizon increases, increasingly more patterns need to be learned from the time series, and the problem that it is difficult for the context vector to express all the patterns and the decoder to fully extract the learned patterns becomes serious. The proposed model connects basic blocks through residuals, so that the patterns that have not been learned by the previous blocks are then passed to the downstream basic blocks to learn. Therefore, the advantage of the proposed model is more obvious when the forecasting horizon increases. As illustrated in Fig. 10, the MAPE of the proposed model is smaller than those of all benchmark models in different horizon scenarios, and the gap increases as the forecasting horizon increases.

F. *Case 2: The Effect of the Decomposition Strategy*

To demonstrate the effectiveness of the decomposition strategy utilized in the proposed model. We compare the proposed model with the TCN encoder-based Seq2Seq model (S2S-TCN), noting that S2S-TCN can be seen as the case where the proposed model has only one basic block. Table IV shows the comparison results of these two models on three datasets under four different forecasting horizon.

TABLE IV
EFFECTIVENESS VALIDATION OF DECOMPOSITION STRATEGY

Prediction Horizon	Model	CH datasets		DE datasets		AT datasets	
		MSE($\cdot 10^{-2}$)	MAPE(%)	MSE($\cdot 10^{-2}$)	MAPE(%)	MSE($\cdot 10^{-2}$)	MAPE(%)
H=3	Proposed	0.0985	2.958	0.0949	4.099	0.102	2.996
	S2S-TCN	0.0974	4.345	0.462	5.007	0.412	4.532
H=6	Proposed	0.143	3.989	0.902	6.900	0.465	4.232
	S2S-TCN	0.276	6.911	1.35	9.336	0.712	6.851
H=12	Proposed	0.225	4.598	1.23	7.416	0.925	5.818
	S2S-TCN	0.414	9.577	1.94	11.418	1.61	6.891
H=24	Proposed	0.343	5.982	2.15	10.524	1.50	8.826
	S2S-TCN	0.587	10.541	3.23	14.792	2.94	13.131

Clearly, the proposed model performs better than the S2S-TCN model. In particular, the advantage of the decomposition strategy is more noticeable when the forecasting horizon increases. For example, in the DE dataset, when the forecasting horizon is 3, the MAPE of the proposed model is 0.907% lower than that of the S2S-TCN, but when the forecasting horizon is increased to 24, the difference between the two MAPEs is 4.27%.

This is because with the decomposition strategy, the proposed model consists of a series of basic blocks and the blocks are linked to each other by residuals, thus dynamically decomposing the time series into multiple components. In this way, the previous basic blocks remove patterns that can be fitted well, allowing the downstream basic blocks to focus more on learning new patterns.

G. Case 3: The Effect of the Number of Basic Blocks

In the proposed model, the number of basic blocks is determined experimentally by the grid search strategy. However, how the number of basic blocks affects the performance of the model remains to be answered. In order to explore the relationship between the model forecast accuracy and the number of basic blocks, comparisons of the proposed model with different numbers of basic blocks on a specific multi-step ahead forecasting task are conducted.

In detail, we build several models with different numbers of basic blocks, i.e., $1, 3, 5, 6, 8$ and 10; and perform independent forecasting on a 12-step ahead forecasting task. These models are denoted as S2S-x, where x represents the number of basic blocks.

According to the obtained results in Table V, as the number of basic blocks increases, the prediction accuracy increases and then levels off. We define the number of basic blocks corresponding to when the accuracy no longer increases significantly as the saturation value. For example, in the CH dataset, when the number of basic blocks is less than 5, the prediction accuracy increases as the number of basic blocks increases. This is because as the number of basic blocks increases, the advantage of the decomposition strategy becomes more obvious, since it can decompose the complex patterns in the time series to be learned separately. After the number of basic blocks exceeds 5, the accuracy remains stable and even starts to decrease slightly because as the number of basic blocks increases, the hypothesis space of the model gradually increases and the model begins to overfit. In this case, the saturation value of the number of basic blocks is 5.

TABLE V
INVESTIGATION OF THE RELATIONSHIP BETWEEN MODEL PREDICTION ACCURACY AND THE NUMBER OF BASIC BLOCKS

Model	CH datasets		DE datasets		AT datasets	
	MSE($\cdot 10^{-2}$)	MAPE(%)	MSE($\cdot 10^{-2}$)	MAPE(%)	MSE($\cdot 10^{-2}$)	MAPE(%)
S2S-1	0.414	9.577	1.94	11.418	1.610	6.891
S2S-3	0.306	6.891	1.54	8.954	0.925	5.818
S2S-4	0.252	5.112	1.43	8.102	0.927	5.821
S2S-5	0.225	4.598	1.37	7.823	0.921	5.782
S2S-6	0.227	4.605	1.23	7.416	0.924	5.81
S2S-8	0.223	4.595	1.22	7.409	0.943	6.143
S2S-10	0.231	4.621	1.29	7.532	0.935	6.108

In addition, it can be found that the saturation value of this hyperparameter varies on different datasets. For example, on the CH dataset, the saturation value is 5; while on the DE dataset the saturation value is 6. This is due to the fact that different time series contain different patterns. The DE dataset contains more complex patterns and therefore requires more basic blocks to fit, which further demonstrates that the DE dataset has relatively large nonsmoothness.

It should be noted that the saturation value varies for different time series, so it is recommended that the grid search strategy be used to determine the saturation value.

H. Case 4: Comparative Study on the Competition Post-COVID-19 Load Demand Data

To further validate the forecasting accuracy and replicability performance of the proposed model, more comparative studies are conducted on the different and specific competition post-COVID-19 load demand datasets (pC19), which consists of historical electricity load data and weather data such as atmospheric pressure, wind speed, temperature, and humidity. Similar to the previous datasets, the post-COVID-19 load demand datasets are preprocessed and split it into training, validation, and test sets. In addition, the mean absolute error (MAE) evaluation metric is used to compare with other participants in the competition in [4], which is calculated as follows:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{15}$$

where y and $\hat{y}$ represent the true value and the forecasted value, and n represents the number of samples.

Table VI presents the corresponding experimental results. From the results, it can be observed that the accuracy of all models decreases as the forecasting horizon increases, which is similar to the previous experimental findings. And the proposed model achieves better performance on different datasets compared to the Naive, MD-XGBoost, and LSTM-LSTM models, with higher accuracy and the increasing advantage as the forecasting horizon expands. When testing on the post-COVID-19 load demand datasets, the accuracy of the proposed model is relatively lower than the top team in [4], since the most teams in [4] combined forecasting from multiple models and utilized a lot of data preparation techniques. Although the ensemble approach of multiple models can achieve a better performance on a specific dataset (i.e., the post-COVID-19 load demand data in [4]), the ensemble approaches need large computational resources and also some other tools for automatic model selection and tuning, which may lead an extra burden to the system operator. Meanwhile, as stated in [4], large ensembles of multiple models can incur significant costs for production and maintenance, as each member of the ensemble requires staff attention and computational effort for relatively minor improvements in forecasting accuracy. Consequently, there are questions regarding whether such models can be justified by system operators due to the additional overhead they create. Additionally, the generality and robustness of the winning models in [4] require further investigation, given that the ensemble methods used a substantial amount of post-COVID-19 data to train their models and it is unclear whether their performance may be adversely affected by different load profile changes brought on by other global or local events.

TABLE VI
EXPERIMENTAL COMPARISON OF THE PROPOSED MODEL ON VARIOUS DATASETS

Prediction horizon	Model	CH datasets	DE datasets	AT datasets	pC19 datasets
H=3	Naive	744.335	4310.464	576.543	51798.929
	MD-XGBoost	446.215	1990.251	281.251	24851.650
	LSTM-LSTM	400.583	1976.135	302.827	22963.095
	Top Team	-	-	-	-
	Proposed	386.416	1589.657	265.298	18156.650
H=6	Naive	956.693	7030.991	928.385	87064.916
	MD-XGBoost	619.568	2412.256	348.624	48910.364
	LSTM-LSTM	632.054	2214.856	316.013	49645.418
	Top Team	-	-	-	-
	Proposed	496.651	1809.734	306.456	31942.453
H=12	Naive	1218.621	10523.884	1388.000	141494.161
	MD-XGBoost	599.423	1951.124	460.101	63218.651
	LSTM-LSTM	540.507	2429.739	423.126	61959.087
	Top Team	-	-	-	-
	Proposed	498.421	1986.328	389.517	38164.458
H=24	Naive	1549.155	10704.823	1404.465	142836.662
	MD-XGBoost	701.651	3515.851	538.291	71852.264
	LSTM-LSTM	658.101	2915.488	546.844	67681.858
	Top Team	-	-	-	10844.000
	Proposed	602.421	2137.161	419.854	42615.284

I. Case 5: Convergence Analysis of Different Forecasting Models

To investigate the training time and convergence rate of different forecasting models, comparative experiments are performed on the same prediction horizon with the same datasets. Specifically, three neural network-based models including the proposed model, S2S-TCN, and LSTM-LSTM are taken to compare their convergence speeds, in which the AT dataset is chosen as the training and testing samples, and the prediction horizon is set to 12. Fig 11 shows the corresponding experimental results, wherein the x-axis represents the model training time and the y-axis indicates the loss value during the training process, and the training time and execution time are given in Table VII. All models are trained and evaluated on the same hardware equipment.

TABLE VII
TRAINING TIME AND EXECUTION TIME COMPARISON BETWEEN PROPOSED MODEL AND MULTIPLE NEURAL NETWORK MODELS

Model	Training Time (s)	Execution Time (s)
Proposed	6757	0.6835
TCN-LSTM	1074	0.5834
LSTM-LSTM	164	0.0772

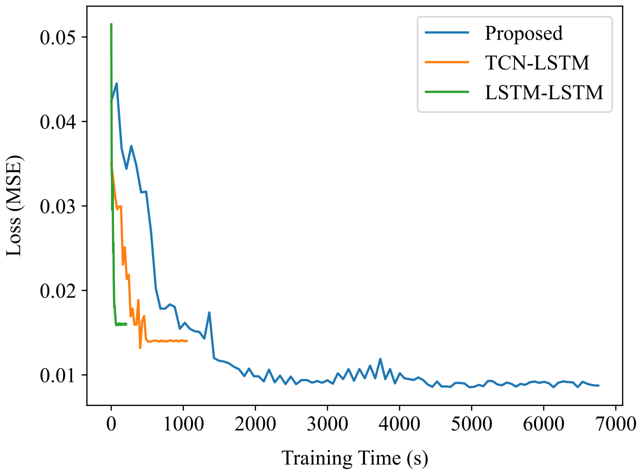


Fig. 11. Training time and model loss comparison between proposed model and multiple neural network models

It can be seen that, the proposed model has slower convergence speed compared to other forecasting models, whereas it achieves lower loss value and higher accuracy upon convergence. This is because the proposed model utilizes the time series addition decomposition strategy, which is able to effectively capture the long-term dependencies and seasonal patterns in the time series data. Moreover, the utilization of multiple basic blocks allows the model to extract more diverse and comprehensive features from the input data, further improving the predictive capability. However, to a certain extent, this resulting in more complex network structures and higher training time in turn, but there is no significant difference on the execution time between the different forecasting models.

J. Case 6: Impact of Sliding Window Size on Model Prediction Accuracy

Generally, an appropriate sliding window size has a significant impact on both the model's prediction accuracy and computational efficiency [46]. To the best of our knowledge, most of the existing studies used fixed or static window size in the sliding window algorithm, and as indicated by [47], there is currently no standard method for determining the value of sliding window size. In this work, a fixed window size is selected based on some empirical guidelines in the literature and a number of preliminary accuracy tests via trial and error. Specifically, on the AT dataset using the MAPE evaluation metric, multiple comparative experiments are conducted to explore the impact of different sliding window size of inputs on the prediction results with different forecasting horizon. Table VIII shows the corresponding experimental results. It can be seen that a larger window size could result in a higher prediction accuracy under a specific forecasting horizon, i.e., when the forecasting horizon H is set to 3, the prediction accuracy is 7.823% when the sliding window size N is chosen to 2, and the prediction accuracy is increased to 2.843% when the sliding window size N is chosen to 72. However, a larger sliding window size usually requires a longer training time.

TABLE VIII
RELATIONSHIP BETWEEN SLIDING WINDOW SIZE AND PREDICTION ACCURACY (MAPE) EXPERIMENT

Windows Size	MAPE(%) for Different Prediction Horizon			
	H = 3	H = 6	H = 12	H = 24
N = 2	7.823	15.644	19.581	21.342
N = 3	7.154	15.716	18.712	22.951
N = 6	4.528	8.006	12.681	20.106
N = 12	2.899	5.091	7.681	16.851
N = 24	2.796	4.232	5.472	9.436
N = 36	2.945	4.594	5.818	8.826
N = 72	2.843	4.119	5.795	8.792

K. Case 7: The Determination Experiments of Encoders and Decoders in the Proposed Model

In this work, a novel multi-step power load forecasting model is designed by incorporating time series additive decomposition strategy into Seq2Seq architecture. This model consists of a series of basic blocks connected through residual, wherein each basic block comprising an encoder and two decoders. For the decoder implementation, the CNN is not considered as an option due to its unsuitability for generating variable-length time series [48]; and the RNN structures such as Gate Recurrent Unit (GRU) and LSTM are already tried and verified in the previous case studies. For the encoder implementation, the CNN, LSTM, GRU and TCN is successively tried in the proposed architecture. Table IX shows the corresponding comparative results on the AT dataset with a 12-step prediction. It can be seen that the best prediction performance is achieved by the TCN-LSTM architecture, thus, the TCN and LSTM are ultimately selected as the encoder and decoder in each basic block of the proposed model.

TABLE IX
COMPARISON EXPERIMENT OF USING DIFFERENT STRUCTURES AS ENCODER AND DECODER

Encoder	Decoder (F-Decoder & E-Decoder)	MAPE(%)	MSE($\cdot 10^{-2}$)
CNN	LSTM	14.518	4.312
	GRU	15.126	4.583
LSTM	LSTM	7.534	1.806
	GRU	7.642	1.901
GRU	LSTM	7.761	2.001
	GRU	7.898	2.146
TCN	LSTM	5.818	0.925
	GRU	6.232	1.045

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a novel Seq2Seq-based deep learning model, which is based on a decomposition strategy combine the Seq2Seq structure, is proposed for multi-step load forecasting. To evaluate the performance of the proposed model, three cases are conducted on three real-world datasets. Case 1 verifies the superiority of the proposed model in terms of accuracy. Specifically, various benchmark models are selected to compare with the proposed model in four scenarios with different forecasting horizons. The results indicate that the proposed model outperforms all benchmark models in terms of accuracy. Furthermore, the performance advantage of the model becomes more obvious as the forecasting horizon increases. Case 2 demonstrates the effectiveness of the decomposition strategy. Case 3 studies the effect of the number of basic blocks on the forecast performance. Case 4 involved an accuracy comparison experiment of the proposed model using the competition post-COVID-19 load demand dataset. Case 5 analyzed the convergence speed of the model during training. In Case 6, a comparison was conducted on the effect of different sliding window sizes on prediction accuracy. Case 7 determined the selection of encoder and decoder for the proposed model through multiple experiments. All experiments prove the performance of the proposed model, so that the model can provide more accurate prediction information for power systems.

For future work, a methodology for adaptively adjusting the number of basic blocks based on the datasets can be developed to further enhance the robustness of the forecasting model and improve the prediction accuracy. Additionally, as the proposed method adopts the MIMO strategy, this may restrict model flexibility and result in biases in the output of the model. Hence, more advanced techniques will be developed to enhance the flexibility of the MIMO strategy.

REFERENCES

[1] C. Zheng, S. Wang, Y. Liu, C. Liu, W. Xie, C. Fang, and S. Liu, "A novel equivalent model of active distribution networks based on lstm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2611–2624, 2019.

[2] G. Dudek, P. Pełka, and S. Smyl, "A hybrid residual dilated lstm and exponential smoothing model for midterm electric load forecasting," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2021.

[3] R. Lu, R. Bai, Z. Luo, J. Jiang, M. Sun, and H.-T. Zhang, "Deep reinforcement learning-based demand response for smart facilities energy management," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 8, pp. 8554–8565, 2022.

[4] M. Farrokhabadi, J. Browell, Y. Wang, S. Makonin, W. Su, and H. Zareipour, "Day-ahead electricity demand forecasting competition: Post-covid paradigm," *IEEE Open Access Journal of Power and Energy*, vol. 9, pp. 185–191, 2022.

[5] M. Khodayar, G. Liu, J. Wang, O. Kaynak, and M. E. Khodayar, "Spatiotemporal behind-the-meter load and pv power forecasting via deep graph dictionary learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4713–4727, 2021.

[6] N. Safari, C. Chung, and G. Price, "Novel multi-step short-term wind power prediction framework based on chaotic time series analysis and singular spectrum analysis," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 590–601, 2017.

[7] J. Li, S. Wei, and W. Dai, "Combination of manifold learning and deep learning algorithms for mid-term electrical load forecasting," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2021.

[8] Y. Yang, W. Li, T. A. Gulliver, and S. Li, "Bayesian deep learning-based probabilistic load forecasting in smart grids," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4703–4713, 2019.

[9] R. Lu, S. H. Hong, and M. Yu, "Demand response for home energy management using reinforcement learning and artificial neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6629–6639, 2019.

[10] C. Li, “Designing a short-term load forecasting model in the urban smart grid system,” *Applied Energy*, vol. 266, p. 114850, 2020.

[11] R. Lu, R. Bai, Y. Ding, M. Wei, J. Jiang, M. Sun, F. Xiao, and H.-T. Zhang, “A hybrid deep learning-based online energy management scheme for industrial microgrid,” *Applied Energy*, vol. 304, p. 117857, 2021.

[12] M. Alhussein, K. Aurangzeb, and S. I. Haider, “Hybrid cnn-lstm model for short-term individual household load forecasting,” *IEEE Access*, vol. 8, pp. 180 544–180 557, 2020.

[13] B. Dietrich, J. Walther, M. Weigold, and E. Abele, “Machine learning based very short term load forecasting of machine tools,” *Applied Energy*, vol. 276, p. 115440, 2020.

[14] S. Tripathi and S. De, “Dynamic prediction of powerline frequency for wide area monitoring and control,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 2837–2846, 2017.

[15] G. Dudek, “Short-term load forecasting using random forests,” in *Intelligent Systems' 2014*. Springer, 2015, pp. 821–828.

[16] R. Lu, Y.-C. Li, Y. Li, J. Jiang, and Y. Ding, “Multi-agent deep reinforcement learning based demand response for discrete manufacturing systems energy management,” *Applied Energy*, vol. 276, p. 115473, 2020.

[17] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[18] V. Singla, S. Singla, S. Feizi, and D. Jacobs, “Low curvature activations reduce overfitting in adversarial training,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 16 423–16 433.

[19] Q. Shi, Y.-M. Cheung, Q. Zhao, and H. Lu, “Feature extraction for incomplete data via low-rank tensor decomposition with feature regularization,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 6, pp. 1803–1817, 2019.

[20] M. Capó, A. Pérez, and J. A. Lozano, “A cheap feature selection approach for the k-means algorithm,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 2195–2208, 2021.

[21] C. Tian, J. Ma, C. Zhang, and P. Zhan, “A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network,” *Energies*, vol. 11, no. 12, p. 3493, 2018.

[22] G. Li, Y. Li, and F. Roozitalab, “Midterm load forecasting: A multistep approach based on phase space reconstruction and support vector machine,” *IEEE Systems Journal*, vol. 14, no. 4, pp. 4967–4977, 2020.

[23] A. Ahmed and M. Khalid, “An intelligent framework for short-term multi-step wind speed forecasting based on functional networks,” *Applied Energy*, vol. 225, pp. 902–911, 2018.

[24] A. Ghasemi, H. Shayeghi, M. Moradzadeh, and M. Nooshyar, “A novel hybrid algorithm for electricity price and load forecasting in smart grids with demand-side management,” *Applied Energy*, vol. 177, pp. 40–59, 2016.

[25] S. B. Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, “A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition,” *Expert Systems with Applications*, vol. 39, no. 8, pp. 7067–7083, 2012.

[26] J. Hännikäinen, “Multi-step forecasting in the presence of breaks,” *Journal of Forecasting*, vol. 37, no. 1, pp. 102–118, 2018.

[27] R. Lu and S. H. Hong, “Incentive-based demand response for smart grid with reinforcement learning and deep neural network,” *Applied Energy*, vol. 236, pp. 937–949, 2019.

[28] J. Deng, X. Chen, R. Jiang, X. Song, and I. W. Tsang, “A multi-view multi-task learning framework for multi-variate time series forecasting,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–16, 2022.

[29] R. Lu, R. Bai, Y. Huang, Y. Li, J. Jiang, and Y. Ding, “Data-driven real-time price-based demand response for industrial facilities energy management,” *Applied Energy*, vol. 283, p. 116291, 2021.

[30] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, “Short-term residential load forecasting based on resident behaviour learning,” *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1087–1088, 2017.

[31] R. Lu, Z. Jiang, H. Wu, Y. Ding, D. Wang, and H.-T. Zhang, “Reward shaping-based actor-critic deep reinforcement learning for residential energy management,” *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 2662–2673, 2023.

[32] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” *arXiv preprint arXiv:1803.01271*, 2018.

[33] E. Skomski, J.-Y. Lee, W. Kim, V. Chandan, S. Katipamula, and B. Hutchinson, “Sequence-to-sequence neural networks for short-term electrical load forecasting in commercial office buildings,” *Energy and Buildings*, vol. 226, p. 110350, 2020.

[34] L. Yang, Q. Wen, B. Yang, and L. Sun, “A robust and efficient multi-scale seasonal-trend decomposition,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5085–5089.

[35] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[36] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, “Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting,” *arXiv preprint arXiv:2202.01575*, 2022.

[37] Q. Wen, J. Gao, X. Song, L. Sun, H. Xu, and S. Zhu, “Robuststl: A robust seasonal-trend decomposition algorithm for long time series,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5409–5416.

[38] Q. Wen, Z. Zhang, Y. Li, and L. Sun, “Fast robuststl: Efficient and robust seasonal-trend decomposition for time series with complex patterns,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2203–2213.

[39] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[40] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, “Deep residual learning in spiking neural networks,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 21 056–21 069.

[41] T. Ahmed, D. H. Vu, K. M. Muttaqi, and A. P. Agalgaonkar, “Load forecasting under changing climatic conditions for the city of sydney, australia,” *Energy*, vol. 142, pp. 911–919, 2018.

[42] Y. Dai and P. Zhao, “A hybrid load forecasting model based on support vector machine with intelligent methods for feature selection and parameter optimization,” *Applied Energy*, vol. 279, p. 115332, 2020.

[43] Y. Chen, P. Xu, Y. Chu, W. Li, Y. Wu, L. Ni, Y. Bao, and K. Wang, “Short-term electrical load forecasting using the support vector regression (svr) model to calculate the demand response baseline for office buildings,” *Applied Energy*, vol. 195, pp. 659–670, 2017.

[44] Y. Wang, S. Sun, X. Chen, X. Zeng, Y. Kong, J. Chen, Y. Guo, and T. Wang, “Short-term load forecasting of industrial customers based on svmd and xgboost,” *International Journal of Electrical Power & Energy Systems*, vol. 129, p. 106830, 2021.

[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[46] Y. Qin, Y. Yan, H. Ji, and Y. Wang, “Recursive correlative statistical analysis method with sliding windows for incipient fault detection,” *IEEE Transactions on Industrial Electronics*, vol. 69, no. 4, pp. 4185–4194, 2022.

[47] H. Gao, C. Liu, Y. Yin, Y. Xu, and Y. Li, “A hybrid approach to trust node assessment and management for vanets cooperative data communication: Historical interaction perspective,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16 504–16 513, 2022.

[48] L. Tang, W. Shen, Z. Zhou, Y. Chen, and Q. Zhang, “Defects of convolutional decoder networks in frequency representation,” *arXiv preprint arXiv:2210.09020*, 2022.

Renzhi Lu (Member, IEEE) received the B.E. in electronic information engineering from the School of Information Science and Engineering, Wuhan University of Science and Technology, Wuhan, China, in 2014, and the Ph.D. degree in electronic systems engineering from the Department of Electronic Systems Engineering, Hanyang University, Ansan, South Korea, in 2019.

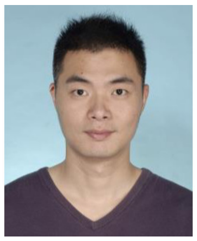
He is currently an Associate Professor with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, where he was a Lecturer from 2019 to 2021. His research interests include learning, optimization, and control with applications to smart manufacturing, smart grid and unmanned system.

Ruichang Bai received the B.E. degree from the School of Automation, HangZhou DianZi University, Hangzhou, China, in 2019, and the M.E. degree from the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, in 2023.

He is currently an Engineer with the Central Academe, Shanghai Electric Group Co., Ltd, China. His research interests include deep learning algorithm design and its application in time-series forecasting.

Ruidong Li (Senior Member, IEEE) received the B.S. degree in engineering from Zhejiang University, China, in 2001, and the M.S. and Ph.D. degrees from the University of Tsukuba, in 2005 and 2008, respectively. He was a Researcher with the Network Architecture Laboratory, National Institute of Information and Communications Technology (NICT).

He is currently an Associate Professor with the Institute of Science and Engineering, Kanazawa University. His current research interests include the Internet of Things, future networks, informationcentric networks, security/secure architectures of future networks, and next-generation wireless networks. He serves on the Editorial Board of IEEE Internet of Things Journal and KSII Transactions on Internet and Information Systems.



Lijun Zhu received the Ph.D. degree in electrical engineering from the University of Newcastle, Australia, Callaghan, NSW, Australia, in 2013.

He is currently a Professor with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China. He was a Postdoctoral Fellow with the Department of Electrical and Electronic Engineering, The University of Hong Kong, and a Postdoctoral Researcher with the University of Newcastle. His research interests include robotics, and nonlinear systems analysis and control.

Mingyang Sun (Senior Member) received the Ph.D. degree from the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., in 2017. From 2017 to 2019, he was a Research Associate and a DSI Affiliate Fellow with Imperial College London.

He is currently a Professor of control science and engineering under the Hundred Talents Program with Zhejiang University, Hangzhou, China. Also, he is an Honorary Lecturer with Imperial College London. His research interests include AI in energy systems and cyber-physical energy system security and control.

Feng Xiao received the B.S. and M.S. degrees in mathematics from Inner Mongolia University, Hohhot, China, in 2001 and 2004, respectively, and the Ph.D. degree in systems and control from Peking University, Beijing, China, in 2008.

He became a Faculty Member with the School of Automation, Beijing Institute of Technology, Beijing, in 2008. He was a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, from June 2010 to May 2013. He was a Visiting Professor with the Department of Mechanical Engineering, University of Victoria, Victoria, BC, Canada, from January 2016 to January 2017. He was also a Professor with the Harbin Institute of Technology, Harbin, China, and is currently a Professor with the School of Control and Computer Engineering, North China Electric Power University, Beijing. His current research interests include group intelligence, coordination control, and networked systems.

Dr. Xiao was a recipient of the Izaak Walton Killam Postdoctoral Fellowship and the Dorothy J. Killam Memorial Postdoctoral Fellow Prize at the University of Alberta in 2010, and a recipient of the Program for New Century Excellent Talents in University, China, and the Excellent Young Scientists Fund by NSFC, China.

Dong Wang (Senior Member, IEEE) received the B.Sc. degree in automation and the M.Eng. degree in control theory and control engineering from the Shenyang University of Technology, Shenyang, China, in 2003 and 2006, respectively, and the Ph.D. degree in control theory and control engineering from the Dalian University of Technology, China, in 2010.

Since 2010, he has been with the Dalian University of Technology, where he is currently a Professor with the School of Control Science and Engineering. His current research interests include multiagent systems, distributed optimization, fault detection, and switched systems. He is an Associate Editor of Information Sciences and Neurocomputing.

Huaming Wu (Senior Member, IEEE) received the B.E. and M.S. degrees in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2009 and 2011, respectively, and the Ph.D. degree in computer science from Freie Universität Berlin, Berlin, Germany, in 2015.

He is currently an Associate Professor with the Center for Applied Mathematics, Tianjin University, Tianjin, China. His research interests include wireless networks, mobile edge computing, Internet of Things, and deep learning.

Yuemin Ding (Senior Member, IEEE) received the Ph.D. degree in electronic systems engineering from Hanyang University, Ansan, South Korea, in 2014.

Since 2021, he has been an Associate Professor with the Department of Electrical and Electronic Engineering, University of Navarra, San Sebastian, Spain. From 2019 to 2021, he was a Postdoctoral Fellow with the Department of Energy and Process Engineering, Norwegian University of Science and Technology, Trondheim, Norway. From 2015 to 2019, he was an Associate Professor with the School of Computer Science and Engineering, Tianjin University of Technology, Tianjin, China. From 2017 to 2018, he was a Visiting Fellow with the Queensland University of Technology, Brisbane, QLD, Australia. His research interests include Internet of Things, communication networks in smart grid, smart homes/buildings, and smart manufacturing.