Applied Soft Computing 144 (2023) 110461

Contents lists available at ScienceDirect

Applied Soft Computing

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

Residual LSTM based short-term load forecasting

Ziyu Sheng [a], Zeyu An [a], Huiwei Wang [a,*], Guo Chen [b], Kun Tian [c]

[a] College of Electronics and Information Engineering, Southwest University, Chongqing 400715, PR China
[b] School of Automation, Central South University, Changsha 410083, PR China
[c] State Key Laboratory of Mountain Bridge and Tunnel Engineering, Chongqing Jiaotong University, Chongqing 400074, PR China

ARTICLE INFO

Article history:
Received 8 December 2020
Received in revised form 11 April 2023
Accepted 18 May 2023
Available online 5 June 2023

Keywords:
Short-term load forecasting
Deep learning
Deep residual network
Long short-term memory

ABSTRACT

As the modern energy systems is becoming more complex and flexible, accurate load forecasting has been the key to scheduling power to meet customers' needs, load switching, and infrastructure development. In this paper, we propose a neural network framework based on a modified deep residual network (DRN) and a long short-term memory (LSTM) recurrent neural network (RNN) for addressing the short-term load forecasting (STLF) problem. The proposed model not only inherits the DRN's excellent characteristic to avoid vanishing gradient for training deeper neural networks, but also continues the LSTM's strong ability to capture nonlinear patterns for time series forecasting. Moreover, through the dimension weighted units based on attention mechanism, the dimension-wise feature response is adaptively recalibrated by explicitly modeling the interdependencies between dimensions, so that we can jointly improve the performance of the model from three aspects: depth, time and feature dimension. The snapshot ensemble method has also been applied to improve the accuracy and robustness of the proposed model. By implementing multiple sets of experiments on two public datasets, we demonstrate that the proposed model has high accuracy, robustness and generalization capability, and can perform STLF better than the existing mainstream models.

1. Introduction

The smart grid has been developed for many years since its inception. As an integral part of the smart grid, load forecasting has also become the object of research by more and more researchers. According to different forecasting purposes, load forecasting is divided into very-short-term load forecasting (VSTLF) [1], short-term load forecasting (STLF) [2], medium-term load forecasting (MTLF) [3] and long-term load forecasting (LTLF) [4]. STLF refers to daily load forecasting and weekly load forecasting, which are used to arrange daily and weekly dispatch plans, including determining the start and stop of generators, water and thermal power coordination, tie-line exchange power, load economic distribution, reservoir dispatch and equipment maintenance, etc. For STLF, it is necessary to fully study the law of power grid load changes and analyze the related factors of load changes, especially the relationship between weather factors, day types, and short-term load changes.

Load forecasting has a very distinctive characteristic, that is, inaccuracy. Since load forecasting is based on the past and present of electric load to speculate its future value, the object of load forecasting work is uncertain events, it is affected by a variety of complex factors, and various influencing factors are also developing and changing. Some of these changes can be estimated in advance, while some are difficult to foresee. Coupled with the impact of changes in some temporary circumstances, the inaccuracy of the forecast results is determined. Compared with the long-term load, the short-term load is flexible and has strong instability, making it more difficult to forecast. However, STLF forecasts the load from one to two weeks in the future. The load in this period has a strong reference value and practicability for energy providers. Therefore, STLF is one of the most hot topics that researchers are concerned about.

Many methods have been proposed to solve STLF problems, which are usually divided into two categories: classic forecasting methods and modern forecasting methods. Widely used classic forecasting methods include linear regression [5], non-linear regression [6], and Autoregressive Integrated Moving Average model (ARIMA) [7]. These methods can improve the accuracy of load forecasting to a relatively high level and are still under continuous improvement and development. Modern forecasting methods include support vector machines (SVMs) [8] and gray data theory [9]. With the development of artificial intelligence and machine learning, artificial neural networks (ANNs) [10,11] have gradually penetrated into various fields and achieved great application results. As novel modern forecasting methods, many network structures based on ANNs that are widely used in STLF have emerged. The most representative ones are convolutional

* Corresponding author.
E-mail address: hwwang@swu.edu.cn (H. Wang).

https://doi.org/10.1016/j.asoc.2023.110461
1568-4946/

neural networks (CNNs) [12] with strong feature extraction capabilities and recurrent neural networks (RNNs) [13] for predicting time series data. They have been shown to perform STLF tasks excellently and have accuracy and generalization capability that many traditional methods cannot reach. In addition to load forecasting, CNN is one of the important methods of image recognition, while RNN plays an important role in the field of natural language processing (NLP) [14]. Based on CNN and RNN, researchers have also proposed many advanced variants: on the basis of CNN, [15] proposed the deep residual networks (DRNs), their biggest feature is the unique shortcut connection, through the shortcut connection, the information in the shallow layers can be transmitted to the deep layers without loss, which largely solves the degradation problem that widely exists in the deep neural networks so that people can train deep neural networks with up to a thousand layers and learn more hidden information; on the basis of RNN, a well-known long and short-term memory (LSTM) [16–19] recurrent neural network has been proposed, compared with RNN, LSTM can selectively memorize and discard information, and solve the vanishing gradient problem in the back propagation when the input is too large. DRN and LSTM are two milestones in the development of neural networks. Their emergence enables neural networks to be better applied to many complex practical problems in various industries, including STLF. [2] applies DRNs to perform STLF and achieves high accuracy, proving the feasibility of using residual networks to solve the STLF problem; [17] uses LSTM to forecast the load of relatively complex individual electric customers, the results show that LSTM also has a good effect in dealing with the STLF problem of time series input. Then, can these two excellent neural network structures be combined for use? This paper will study and discuss this topic.

In this paper, our main contribution is proposing a neural network framework that combines DRN and LSTM to perform STLF. We know that LSTM can selectively discard and retain the previous information when processing the time series input, so as to make better and more flexible use of intermediate information; DRNs can transfer the information in the shallow layers to the deep layers through shortcut connections, although DRNs are not able to selectively memorize information, the idea of information transmission has great similarities with LSTM. Inspired by this, we try to apply the combination of DRN and LSTM into STLF problems and present our proposed model:

(1) First of all, we adopt ResNet Plus as the overall network framework, it is a modified residual network variant based on the DRN. The results in [2] show that compared to ordinary residual networks, ResNet Plus has higher accuracy in STLF, we will introduce it in detail later.

(2) A residual network is composed of many stacked residual blocks, and the structure in the block can be set differently depending on the task. For example, when performing image-related tasks, convolutional layers and pooling layers can be added to the block; and when time series data forecasts are required, we often use the RNN structure. In our proposed model, we apply LSTM layers into the residual blocks. Each residual block contains two LSTM layers. DRN-specific shortcuts are also added between the LSTM layers. We use this residual block with LSTM structure to forecast the time series input data.

(3) It is worth mentioning that we add a calculation unit called dimension weighted unit (DWU) [20,21] after each LSTM layer to explicitly model the interdependencies between feature dimensions. Subsequent experiments will prove that the dimension weighted unit can help us significantly improve the performance of our proposed model.

(4) Moreover, we use dropout [22] and batch normalization (BN) [23] to regularize the data to prevent overfitting, and the ensemble method is also adopted to improve the robustness of our proposed model. We name the proposed model Residual LSTM Plus. This model combines the characteristics of DRN, LSTM, and dimension weight units to improve the performance of the network from three aspects: depth, time, and feature dimensions.

(5) In the experimental part, we compare the accuracy of the model under different structures and hyperparameter settings, and demonstrate the specific performance of the model in different periods. We also compare our proposed model with several mainstream models. The results prove that our proposed Residual LSTM Plus has state-of-the-art performance.

The rest of the paper is organized as follows. In Section 2, we give a detailed introduction to our proposed Residual LSTM Plus model structure and the background knowledge involved. The dimension weighted unit, activation function, ensemble methods and model implementation details are also be provided. In Section 3, we discuss and analyze the impact of model structure and hyperparameter settings on model performance, compare our model with the existing models on two datasets, and the forecasting performance of our proposed model is verified over multiple periods. Section 4 summarizes this paper and proposes future improvements and research directions.

2. STLF based on Residual LSTM Plus

In this section, we propose a neural network model based on DRN and LSTM called Residual LSTM Plus, which is used to perform STLF on the electrical load within 24 h of the next day. The model has two parts. The first part is a fully-connected neural network. The inputs are feed into this network to get a preliminary prediction of the load for the next 24 h. Next, the forecast results of the first part are transmitted to the second part of our model, the residual LSTM network, for further forecasts to get more accurate results. In addition, our proposed model uses many cutting-edge methods such as dimension weighted unit, snapshot ensemble method [24], BN, dropout, etc. to improve the accuracy and robustness of the model. In the following subsections, we will first introduce the overall framework of the model, and then introduce the specific design of each part in detail.

2.1. Residual LSTM Plus

In this subsection we introduce the overall framework of the proposed model Residual LSTM Plus. The model has two parts, the first part is a fully-connected neural network proposed in [2]. Through this fully-connected neural network, we obtain a preliminary prediction of the load in the next 24 h. Its structure is shown in Fig. 1, the structure shown in the figure is a sub-network, and each individual sub-network predicts the load of the corresponding hour in the future. Finally, the 24 sub-networks are integrated to obtain the preliminary forecast value of the load in the next day. L_m and T_m respectively represent the power loads and temperature values of the same period that are 1, 2, and 3 months before the next day; L_w and T_w respectively represent the power loads and temperature values of the same period that are 1 to 8 weeks prior to the next day; L_d and T_d respectively represent the power loads and temperature values of the same period of everyday of the recent week prior to the next day; L_h represents the power loads within 24 h before the next day; T represents the temperature value in the next day. In practice, the difference between the predicted temperature value obtained through the weather forecast and the actual temperature value may have a certain impact on the accuracy of the model; S, W, and H respectively represent the season, weekday and holiday using one-hot code. These inputs are obtained by preprocessing the time, temperature and load data recorded in the dataset.

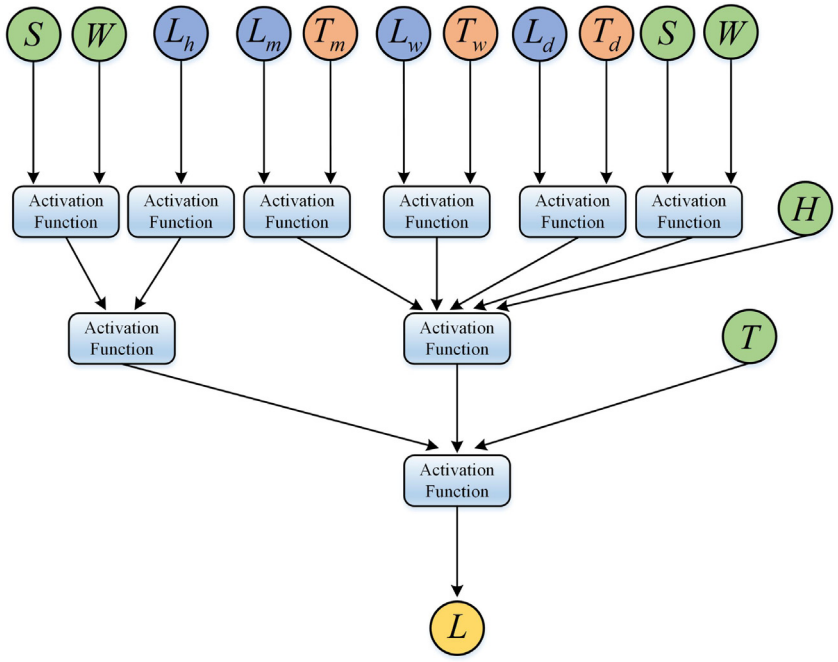


Fig. 1. The first part of our proposed model: fully-connected neural network. In this paper, the activation function is SELU.

These inputs are then sent to the fully-connected neural network, and the preliminary forecast results L of the load are obtained. The preliminary forecast results are not very accurate, so next they are then passed through our second part residual LSTM network, which is a network composed of stacked residual LSTM blocks for further forecasts to improve accuracy.

The second part of our proposed model uses the modified residual network architecture ResNet Plus, and the network is composed of many stacked residual blocks, so we first introduce the block structure within a residual LSTM block. As shown in Fig. 2, the residual LSTM block is composed of LSTM, dimension weighted unit, BN and dropout, its core is two LSTM layers. First, the input x_l of the residual block is normalized by BN to a value with a mean of 0 and a variance of 1, then it is passed through the first LSTM layer, the LSTM output obtained is randomly deactivated through the dropout layer, the neuron will be randomly discarded according to the set probability value, and the ongoing operations on the node will be cleared, and finally a simplified neural network is obtained. In our proposed model, dropout and BN are used together to play a regularization role to prevent overfitting. Subsequently, the interdependencies between feature dimensions is learned by the dimension weighted unit and the features are updated according to the weights. The output of the dimension weighted unit is added with the shortcut connection to get $\tilde{x}_l$, $\tilde{x}_l$ performs the same operations again, and we obtain the final output x_{l+1} of the residual LSTM block. These residual LSTM blocks are stacked end to end to form the complete network. The overall framework of our proposed model can be seen in Fig. 3, the preliminary forecasts of the fully-connected network are passed to the followed residual LSTM network for further prediction, which greatly improves the accuracy and generates the final output. The Residual LSTM Plus framework combines the advantages of ResNet Plus, LSTM and Squeeze-and-Excitation Network, which can flexibly and fully utilize useful information in the time dimension and feature dimension; through denser shortcut connections, data can be transmitted from the shallow network to the deep network unimpeded, the efficiency of back propagation is improved and features are fully reused; it also well solves the vanishing/exploding gradient and degradation

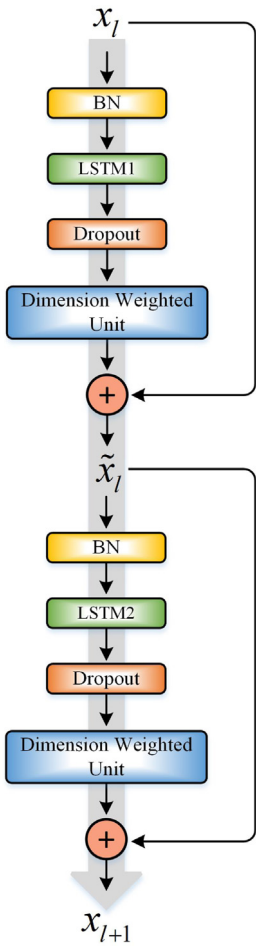


Fig. 2. Residual LSTM block structure.

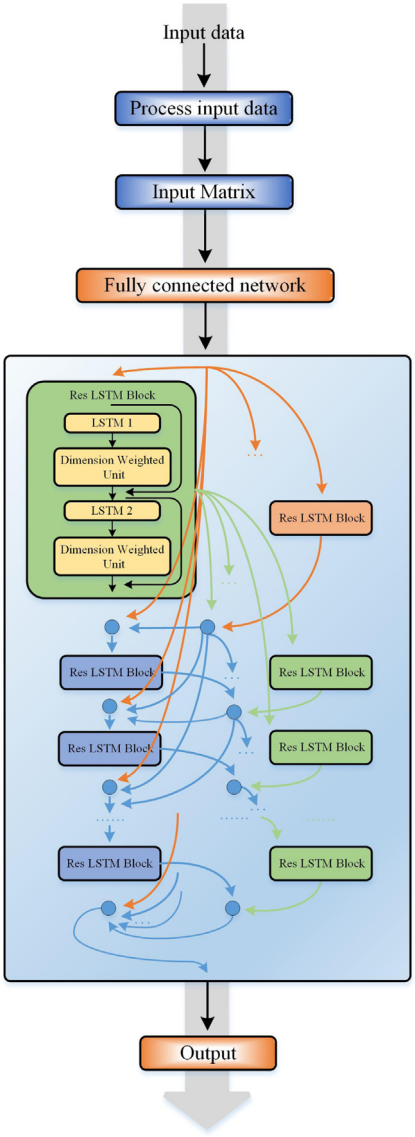


Fig. 3. The Residual LSTM Plus based STLF framework.

problem. In the subsequent subsections of Section 2, a detailed introduction to each substructure of the proposed model Residual LSTM Plus is given.

2.2. ResNet Plus

STLF is a task with high requirements for accuracy, so the main goal of current research is to improve the accuracy of the model. We know that there exist many methods for neural networks to improve the accuracy of predictions, such as modified network architectures [25], parameter optimization [26], data augmentation [27] and many other methods can improve the performance of the model to a certain extent, another direct and effective method is to increase the depth of the network. Previous research results show that, in theory, whether it is dealing with regression problems or classification problems, compared with shallow networks, deep neural networks can learn more abstract features and hidden structures to further improve the accuracy of the network. However, in practical applications, the increase in network depth will bring about a series of problems, such as vanishing/exploding gradient and degradation problem that are prone to occur in deep networks. With the invention of the ReLU activation function, the vanishing/exploding gradient has been well solved, but the degradation problem has plagued researchers for a long time, some of them even sigh: why is it hard to train deep neural networks? Degeneracy, not vanishing gradients, is the key. Briefly speaking, the degradation problem refers to the phenomenon that as the network depth increases, the accuracy of training appears to be saturated. If degradation problem occurs, the performance of the deep neural network may even be worse than the shallow neural network. Many studies [28,29] show that the degradation problem seriously affects the training of neural networks, the cause of degradation problem is discussed in [30]: Due to the existence of the nonlinear activation function, the process from input to output is almost irreversible, which will cause a certain degree of information loss. As the depth of the model continues to increase, the loss of this information, especially useful information on some important features, will accumulate, making the performance of the model difficult to be equal to before.

In order to solve the degradation problem, people have also proposed many methods. For example, [31] proposes a method of calculating the entire singular value distribution of the Jacobian matrix of a nonlinear network to solve the degradation problem. Among the solutions to the degradation problem, the deep residual network (DRN) proposed in [15] is the most famous. A DRN is composed of many stacked residual blocks, as shown in Fig. 4, compared with ordinary neural networks, DRNs have identity shortcut connections that connect the input and output, which are the core feature of the DRNs. Through the shortcut connections, the DRNs have the ability of identity mapping. When the number of network layers increases, it is ensured that the performance of the deep network is at least not inferior to the shallow neural network. This is the original intention of the DRNs and the method to solve the degradation problem. A standard residual block can be expressed as:

$$\mathcal{H}(x) = \mathcal{F}(x, W_i) + x, \tag{1}$$

where x and $\mathcal{H}(x)$ are the input and output of the residual block respectively, W_i is the weight parameter, and $\mathcal{F}(x, W_i)$ is the residual we want to learn. Taking Fig. 4 as an example, the residual is the double-layer weight activated by the activation function, we move x to the left side of the original residual block equation and get the residual function:

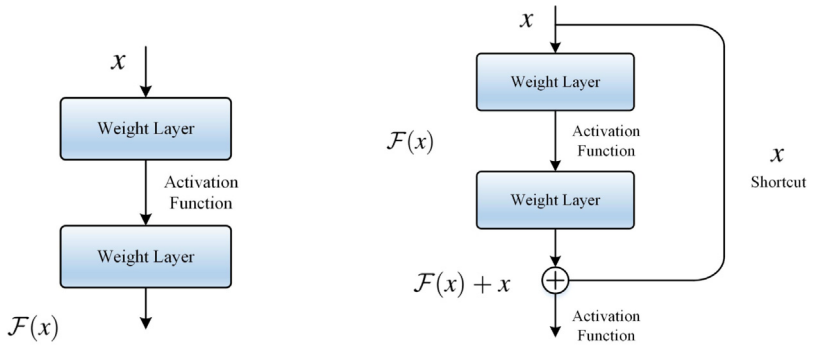
$$\mathcal{F}(x) = \mathcal{H}(x) - x. \tag{2}$$

In this form, our problem is transformed into the learning of the residual function, and the amount of calculation of the learning residual is relatively small, which allows the model to converge faster. Moreover, during back propagation, we can get the derivative of $\mathcal{H}(x)$:

$$\mathcal{H}'(x) = \mathcal{F}'(x) + 1, \tag{3}$$

the constant **1** represents the shortcut connection and it ensures that the gradient can be transmitted intact, which largely solves the vanishing gradient problem, making it possible to train very deep neural networks.

After [15], researchers have further explored the residual networks and proposed many novel residual network variants, such as DenseNet [32], which maximizes the use of shortcut connection, and establishes a shortcut connection between every two layers; DRN and DenseNet are further used as dual channels for parallel processing, the results are merged to get dual path network (DPN) [33], etc. These new residual network structures not only improve the accuracy rate, but also increase the generalization capability of the model so that the residual network can be applied to various practical problems.



(a) Ordinary block (b) Residual block

Fig. 4. The comparison of ordinary neural network and residual network.

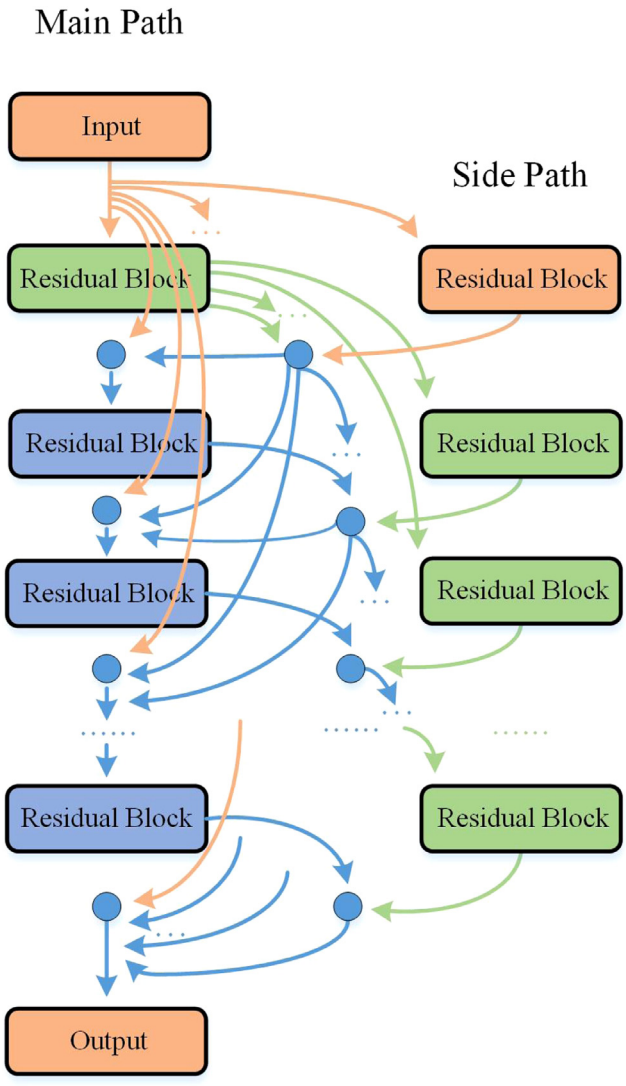


Fig. 5. ResNet Plus network structure. (The blue dots average their inputs).

In [2], the residual network structure was optimized and redesigned, a DRN-based residual network variant ResNet Plus was proposed and successfully applied to STLF, proving that the structure can be well applied to nonlinear regression problems. Therefore, our proposed model also chooses this network structure and combines it with LSTM to achieve better results than the original model ResNet Plus. The network architecture of ResNet Plus is shown in Fig. 5. Different from the typical residual network, ResNet Plus adds the side residual blocks on the right, so that there are two paths in the network, of which the main path is on the left. In the side path, except for the input of the first residual block which is the input of the entire network, the input of the other side residual blocks is the output of the first residual block in the main path. In each layer, the output of the residual block on the main path will be averaged with the output of the side residual block (the blue dots in Fig. 5 average their inputs), and its result is connected to all subsequent residual blocks on the main path. In other words, except for the first layer, each residual block on the main path is densely connected to all residual blocks prior to the layer in the network, which is somewhat similar to the idea of DenseNet [32].

We can see that compared to the traditional residual network, ResNet Plus has more residual blocks and denser shortcut connections, which greatly improves the efficiency of back propagation, increases the reuse of features, and can conveniently transmit the information of the shallow network to the deep layer of the network through shortcuts. Cooperating with the LSTM layers inside the residual block, the information can be better utilized.

2.3. LSTM

Recurrent neural networks (RNNs) are special sequence-based neural networks that can process time series input. For ordinary neural networks, the previous input will not affect the subsequent input, and the data are independent of each other. Different from ordinary neural networks, RNNs retain the input information and pass it to the next input, which is equivalent to our networks having the ability to remember information, and the input of the previous part can affect the input of the latter part. However, the traditional RNNs also have many problems: RNNs record the input data, including a lot of useless information, which may largely increase the network size; RNNs exist the long-term dependencies problem [34], that is, the relevant information is too far away from the current information, when the distance is very large, the relevant information cannot be learned; the sigmoid function is used as the activation function, which may cause vanishing/exploding gradient [35].

In order to overcome the above issues in RNN, the long short-term memory (LSTM) was proposed in [16]. By adding an extra gate mechanism such as forget gate and memory gate on the basis of RNN, the network can selectively forget part of information and make full use of current information. The LSTM architecture is shown in Fig. 6. The figure shows the basic LSTM unit called

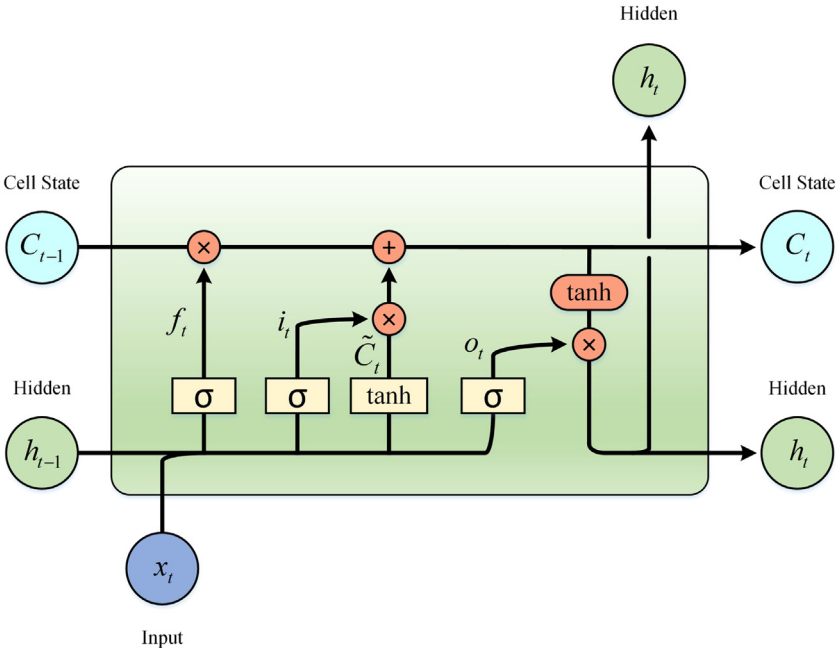


Fig. 6. The architecture of an LSTM cell.

cell, and an LSTM is composed of many connected cells. Compared with RNN, the characteristic of LSTM is the cell state throughout the whole life cycle of LSTM which can be seen at the top of Fig. 6, it is also the core parameter of LSTM. We use $\{x_1, x_2, \ldots, x_t\}$ to denote a typical time series input for LSTM, of which x_t represents the input value at the tth time step. As shown in Fig. 6, by interacting with the input x_t and the intermediate output h_{t-1}, the cell state C_{t-1} can determine which elements should be forgotten or updated. The first step in LSTM is to determine what elements to forget, this is done through the sigmoid layer called forget gate. The specific implementation is to linearly combine the input x_t and the intermediate output h_{t-1} (b is the bias term), and then obtain f_t through the activation function σ, which can be expressed as:

$$f_t = \sigma\ (W_f \cdot [h_{t-1}, x_t]\ +\ b_f). \tag{4}$$

Since sigmoid is used as the activation function, the f_t obtained is a real number between $(0, 1)$, then multiply it with the cell state C_{t-1} to determine which elements need to be erased. After determining the erased elements, we then need to obtain the retained elements and use the new data to update the cell state. We combine the input x_t with the intermediate output h_{t-1} and get i_t through the activation function σ. i_t represents the new elements we want to keep, which can be shown as:

$$i_t = \sigma\ (W_i \cdot [h_{t-1}, x_t]\ +\ b_i). \tag{5}$$

After getting i_t, we need to update the cell state through new elements. First, we linearly combine x_t and h_{t-1}, and then perform nonlinear transformation through tanh function to obtain the cell state $\tilde{C}_t$ formed by the elements. Its formulation can be expressed as:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t]\ +\ b_C). \tag{6}$$

The newly obtained cell state $\tilde{C}_t$ is multiplied by i_t, and then added to the original cell state, we finally get the updated new cell state C_t, which determines what elements are forgotten and what elements are retained, which can be expressed for:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{7}$$

where $\odot$ represents the element-wise multiplication. After updating the cell state, the final output of this cell should be determined, which is based on the current cell state. The output of this cell can be divided into two parts. The first part is the cell state C_t, this parameter needs to be updated and maintained throughout the whole life cycle of the LSTM, so it needs to be sent to the next cell; the second part is the intermediate output h_t between the cells, it represents the information we need to keep and pass to the next cell. We first combine h_{t-1} and x_t and get o_t through the activation function σ, the expression is as follows:

$$o_t = \sigma\ (W_o \cdot [h_{t-1}, x_t]\ +\ b_o). \tag{8}$$

Then the updated cell state C_t is normalized between $(-1, 1)$ through the tanh function, and multiplied with o_t to get the output of this LSTM cell h_t after we forget and retain the elements:

$$h_t = \tanh(C_t) \odot o_t. \tag{9}$$

When this cell is the last cell in the LSTM, we only need to output h_t as the final result of the LSTM prediction. In view of the excellent performance of LSTM, we have added an LSTM layer to each residual block in our proposed model. LSTM can capture the subtle hidden relationship between the previous input and the current input in the time dimension and provide us with more accurate load forecasts.

2.4. Dimension weighted unit

We know that LSTM can forget and retain information in the time dimension, but it does not perform similar calculations on the feature dimensions of hidden nodes, so the output features of LSTM do not make full use of the useful information between the feature dimensions of the hidden nodes. Therefore, we have applied attention and gating mechanisms to our proposed model. The attention mechanism can bias the allocation of available processing resources to the most valuable information in an input semaphore, it is generally used for the adjustment of higher-dimensional abstract information expression in one or more layers among multiple modules. Recent years, in the field of image and lip language, attention mechanism has been applied

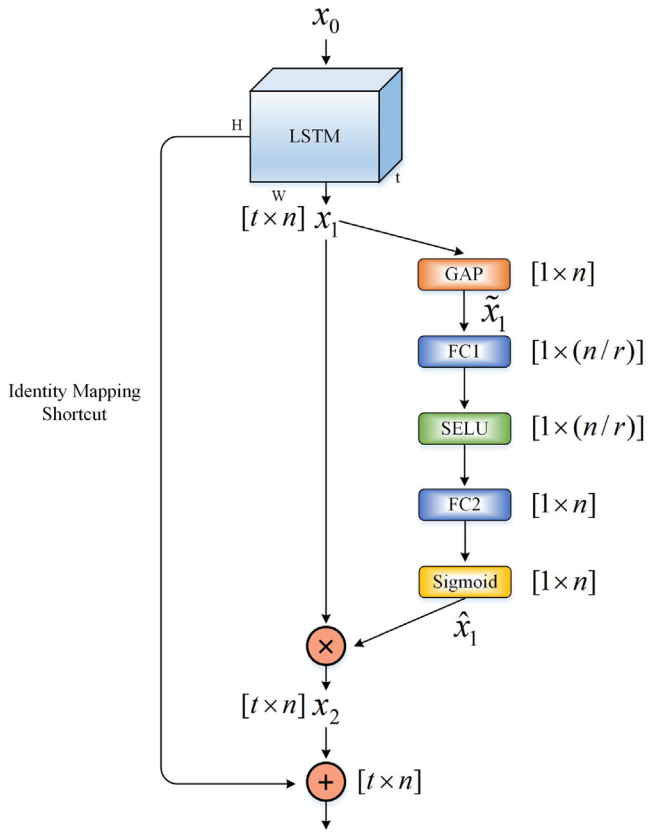


Fig. 7. The dimension weighted network structure.

to efficiently aggregate multi-module data, which also shows its applicability in specific fields. Attention mechanism is usually realized through gating mechanism. We add a dimension weighted unit (DWU) after each LSTM layer, this structure is based on the Squeeze-and-Excitation Network discussed in [20,21], it is a lightweight gating mechanism that can explicitly model the interdependencies between feature dimensions in an efficient calculation method and can be inserted into any position in the network to enhance the expressive ability of the model. Thus, the network can selectively amplify valuable features and suppress useless ones from the perspective of global information.

The dimension weighted network structure is shown in Fig. 7. We can see that the dimension weighted unit is composed of a global average pooling layer (GAP), fully connected layers (FC), and nonlinear activation functions. According to the steps, it can be divided into two operations, squeeze and excitation. x_0 is the input of the dimension weighted network; t represents different periods in the time series; n is the output dimension of the LSTM layer, its size is $H \times W$; x_1 is the output of the LSTM layer (For ease of understanding, we represent the three-dimensional vector whose original size is $[H \times W \times t]$ as a two-dimensional vector $[t \times n]$); x_2 is the updated feature after the LSTM layer output is multiplied by the dimension weighted unit. First, the input x_0 passes through the LSTM layer, and the two-dimensional feature vector x_1 of size $[t \times n]$ is output. Next, we use the global average pooling layer to compress the global features of $[t \times n]$ into a two-dimensional vector $\tilde{x_1}$ of size $[1 \times n]$ according to the time dimension, that is, each dimension in t periods is compressed into a real number, this real number represents the global distribution of the original feature in the time dimension, and has a global receptive field. $\tilde{x_1}$ can be expressed as:

$$\tilde{x_1} = \frac{1}{t}\sum_{i=1}^{t} x_1^i. \tag{10}$$

Through the squeeze operation, we can extract the main features in x_1, expand the receptive field, and greatly reduce the computational cost. Subsequently, we explicitly model the interdependencies between feature dimensions when the information is transmitted from the LSTM layer to the next layer, evaluate the weights of the features and output the weights. This is the excitation step, which is carried out by the two fully-connected layers FC1 and FC2 and their activation operations. The feature dimension of FC1 is $\frac{1}{r}$, r is the hyperparameter we set to adjust the dimension, which represents the multiple of dimension reduction, it is an important hyperparameter that can change the size and computational cost of the dimension weighted unit in the model. In the experimental part, we compare the performance of the model with different r values to obtain the best hyperparameter r. The activation function of FC1 is selu, and the size of data after passing through FC1 is $[1 \times \frac{n}{r}]$. FC2 restores the dimension, its activation function is sigmoid. Through the sigmoid function, the result is normalized to a weight value between (0, 1), and the output $\hat{x}_1$ of the dimension weighted unit is obtained:

$$\hat{x}_1 = \sigma(W_2 \cdot \delta(W_1 \cdot \tilde{x_1})), \tag{11}$$

where δ represents the nonlinear activation function selu; σ represents the sigmoid function; W_1 and W_2 are the weight parameters of the fully-connected layers FC1 and FC2, respectively. Finally, we multiply the original feature of the LSTM output x_1 with the weights we learned through the dimension weighted unit to get the updated feature x_2, and then the shortcut connection is added to x_2.

In summary, the dimension weighted unit can fit the complex interdependencies between feature dimensions through the squeeze-excitation operation. By adding a small number of parameters, we can learn the weight of each feature dimension when information is transmitted between network layers, and through weight rewards and punishments we can enhance the features of important dimensions, suppress the features of useless dimensions, strengthen the feature expression ability of the model. After applying the dimension weighted unit, the performance of our proposed model is further improved at the expense of a small amount of calculation. In the experimental part, we test and discuss the actual application performance of the dimension weighted unit.

2.5. Ensemble method

In the previous sections, we present the Residual LSTM Plus framework and focus on how to improve the accuracy of our proposed model. For a long time, researchers have often regarded accuracy as the core or even the only criterion for evaluating model performance. MIT and IBM research institute have conducted research on the robustness of 18 widely used deep neural network models and demonstrated their research results in [36]. The results show that even a well-trained deep neural network model can be easily broken by adversarial attack algorithms. The target model often misclassifies the adversarial samples after adding noise. When the model only pursues higher accuracy, a certain degree of robustness is often sacrificed. Therefore, robustness should also be one of the important properties considered by our proposed model. The experimental results in [37] express that the ensemble method can not only improve the test accuracy, but also improve the robustness of the network. Therefore, we apply

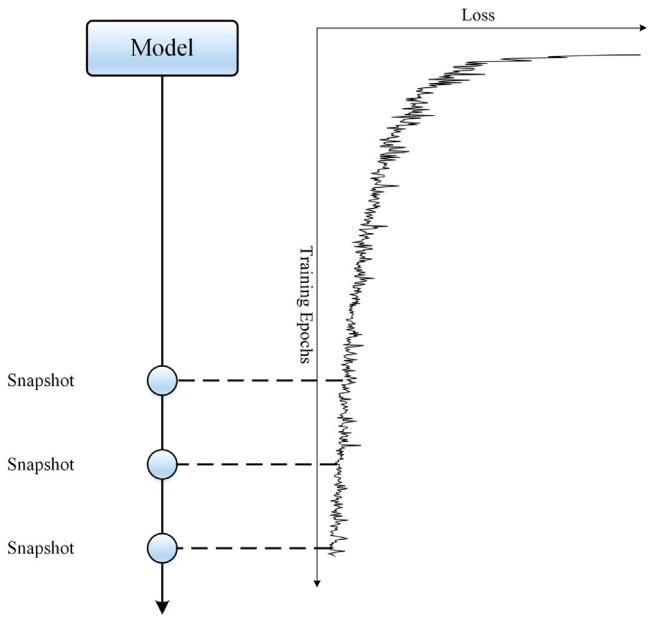


Fig. 8. Snapshot ensemble method.

the snapshot ensemble method [2,24] shown in Fig. 8 into the proposed model: several snapshots of the same model are taken after an appropriate number of epochs, at this time, the loss of each snapshot is at a similar and low level. The models at these snapshot points are saved, we then ensemble these models to get the final output. The advantage of the snapshot ensemble method is that it can train multiple models in a single training time, which greatly reduces the training time and computational cost. We use Adam as the optimizer, the learning rate schedule can be adjusted adaptively without setting, so this is an ensemble method similar to NoCycle snapshot in [24] that does not use periodic learning rate scheduling. Through the snapshot ensemble method, we can ensemble multiple models with relatively small errors in the same training time, increase the robustness of the model without sacrificing accuracy, and greatly improve the performance of our proposed model.

2.6. Model design and implementation details

In this paper, we propose a neural network model Residual LSTM Plus based on DRN, LSTM, and Squeeze-and-Excitation Network for STLF of the load in the next 24 h. The framework of our proposed model is shown in Fig. 3. The model has two parts, the first part is a low-level fully-connected neural network proposed in [2], which can obtain preliminary predictions in the next 24 h by processing the input. The second part of the model is our redesigned residual LSTM network. We can obtain more accurate forecasts by further predicting the results of the first part. In the network structure, we adopt a modified residual network structure ResNet Plus. By adding side residual blocks and denser shortcut connections, data can be transmitted more smoothly in the model, the feature reusability, expressive ability and the back propagation efficiency of our proposed model have been improved. In each residual block, we add LSTM (each LSTM contains 256 hidden units) layers and dimension weighted units, so that the model can better learn the hidden relationship in time and feature dimension. The snapshot ensemble method is also applied to improve the robustness and accuracy of the model. We use batch normalization (BN) and dropout as regularization methods to reduce overfitting, the dropout rate is set to 0.5 because the model generates the most hidden structures at this rate.

The activation function is an important part of the neural network, which can perform nonlinear transformations on the input. A variety of different activation functions are also applied to our model: in the LSTM and dimension weighted unit, we use the sigmoid function in the form as follows:

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}. \tag{12}$$

The sigmoid function can map the input to the interval of $(0, 1)$, it is used as a gate unit in LSTM and dimension weighted units. In LSTM, we also use the tanh function to form new cell states, its formula can be expressed as:

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1. \tag{13}$$

The tanh function can map the input to the $(-1, 1)$ interval, it solves the problem that the output of the sigmoid function is not zero-centered, but the vanishing/exploding gradient still exists. The activation function most used in our model is the scaled exponential linear unit (SELU) function [38], its formula is:

$$\text{SELU}(x) = \lambda \begin{cases} x, & \text{if } x > 0 \\ \alpha e^{x} - \alpha, & \text{if } x \le 0, \end{cases} \tag{14}$$

where λ and α are the values derived from [38], $\lambda \approx 1.0507$ and $\alpha \approx 1.6733$. SELU is a modified version of ReLU [39]. The ReLU function is widely used in various neural network models, but it also has a defect: when a large gradient passes through a ReLU neuron, this neuron may not be activated by any input, this problem is also called dead neuron. The negative axis of SELU is non-zero, which can solve the dead neuron problem well; SELU also has self-normalization properties, which can approximate the excitation value to 0 mean and unit variance so that it can help the neural network to avoid the vanishing/exploding gradient and enhance the robustness of the model. Different from ordinary activation functions, the weights of the hidden layers need to be initialized by LeCun normal initializer when using SELU, and the network structure must be sequential. In this paper, SELU is used in two positions: the fully-connected neural network shown in Fig. 1 and the dimension weighted unit shown in Fig. 7.

The cost function of our proposed model is as follows:

$$\mathcal{L} = \mathcal{L}\text{mape} + \mathcal{L}_{\text{penalty}}, \tag{15}$$

it is formulated as the sum of two parts. $\mathcal{L}_{\text{mape}}$ is the error measure term, which uses mean absolute percentage error (MAPE). In this paper, MAPE is not only part of the cost function, but also used to measure the performance of the model in the experimental part, its form can be expressed as:

$$\mathcal{L}\text{mape} = \frac{1}{NH}\sum_{i=1}^{N}\sum_{h=1}^{H}\frac{\left|\text{obs}_{(i,h)} - \text{pred}_{(i,h)}\right|}{\text{obs}_{(i,h)}}, \tag{16}$$

where $\text{obs}_{(i,h)}$ represents the actual observed load value in the grid at the hth hour of the ith day; $\text{pred}_{(i,h)}$ is the predicted value of the model in the same period; N is the number of samples; H is the number of hourly loads in a day, in our case $H = 24$. $\mathcal{L}_{\text{penalty}}$ can be formulated as:

$$\mathcal{L}_{\text{penalty}} = \frac{1}{2N}\sum_{i=1}^{N}\left[\max\left(0, \max_{h}\text{pred}_{(i,h)} - \max_{h}\text{obs}_{(i,h)}\right) + \max\left(0, \min_{h}\text{obs}_{(i,h)} - \min_{h}\text{pred}_{(i,h)}\right)\right], \tag{17}$$

it is a penalty term that can punish the model when our forecast value deviates from the true value, and when our prediction error is small, this term will also emphasize the cost of overestimating the peak and valley of the load curve. This cost function was proposed in [2]. Compared with the traditional cost function, this cost function is more suitable for STLF problems.

In order to test the fitting degree of the forecasting results of the proposed model to the actual sample data on the test set of different years, we also use a statistical-based model performance evaluation method: the coefficient of determination (R^2) to measure the performance of the model, it can be expressed as:

$$R^2 = \frac{\left[\sum_{i=1}^{N}\sum_{h=1}^{H}\left(\mathrm{obs}_{(i,h)} - \overline{\mathrm{obs}}\right)\left(\mathrm{pred}_{(i,h)} - \overline{\mathrm{pred}}\right)\right]^2}{\sum_{i=1}^{N}\sum_{h=1}^{H}\left(\mathrm{obs}_{(i,h)} - \overline{\mathrm{obs}}\right)^2 \sum_{i=1}^{N}\sum_{h=1}^{H}\left(\mathrm{pred}_{(i,h)} - \overline{\mathrm{pred}}\right)^2}, \tag{18}$$

where $\overline{\mathrm{obs}}$ and $\overline{\mathrm{pred}}$ are the mean of observed and predicted load data. The closer R^2 is to 1, the better the model fits the data sample.

3. Experimental results and analysis

In this section, we verify the performance of the proposed Residual LSTM plus model through multiple sets of experiments. First, we introduce the dataset used in the experiment and its data distribution; based on these datasets, we then compare the performance of the model with different hyperparameter settings and structures; subsequently, we compare the model with the existing mainstream models and proved that our proposed model has excellent performance among similar models; finally, we compare and discuss the performance of the model in different years and seasons.

Multiple approaches are taken to ensure the fairness of the comparison experiments. First, we use consistent hyperparameter settings including the number of iterations, batch size, and optimizer with reference to [2]. Moreover, we conduct comparison experiments on two different real-world datasets: ISO-NE and Malaysia to verify the effectiveness and generalization capability of the models. We also test the models on load data of different years and a statistical-based method R^2 is used to evaluate the performance.

3.1. Dataset introduction and hyperparameter settings

Our experiments are based on the ISO-NE dataset[1] and the Malaysia dataset[2] [40]. The ISO-NE dataset records the hourly load data of 4324 days in the New England area of the United States from 2003 to 2014, the total sample size is 103,776; the Malaysia dataset records the hourly load data of the power company in the city of Johor, Malaysia for 730 days from 2009 to 2010, the total sample size is 17,520. The load values recorded in these two datasets are shown in Fig. 9. We selected datasets of different regions and different sample sizes to verify the generalization capability of our model. The ISO-NE dataset is the default dataset, and most of the benchmark experiments are run based on it, while the Malaysia dataset is used in the comparison experiment with existing mainstream models. The division of training set and test set in the two datasets is shown in Table 1. The ISO-NE dataset uses hourly load data from May 2003 to December 2005 as the training set, and the test set uses the load data of the following year; the Malaysia dataset uses the load data of the last two months as the test set and the rest as the training set.

Table 1
Division of training set and test set in ISO-NE and Malaysia datasets.

Dataset		Training set	Test set
ISO-NE	Start from	05/24/03 01:00 a.m.	12/31/05 01:00 a.m.
	To	12/30/05 12:00 p.m.	12/30/06 12:00 p.m.
Malaysia	Start from	01/01/09 01:00 a.m.	11/01/10 01:00 a.m
	To	10/31/10 12:00 p.m.	12/31/10 12:00 p.m.

Table 2 shows the hyperparameter settings of multiple mainstream neural network models we use for comparison. In order to ensure the fairness of the experiments, we refer to the settings of ResNet Plus in the model's hyperparameters and optimizer, and use the same hyperparameter settings on all datasets to verify its generalization performance. These compared models use the Adam optimizer and snapshot ensemble method, and the number of training iterations and batch size are consistent with our proposed model. Each training process of the model requires 700 iterations. When the model is trained for about 700 epochs, its loss is relatively low and tends to be stable, which shows that the optimization algorithm has converged and is close to the global optimum. When we continue to train the model on the basis of 700 iterations, the decrease of loss will be very limited and the training time will increase. Therefore, after many experiments, we determined that 700 is the appropriate number of iterations. We apply the snapshot ensemble method into the model, and three snapshots are taken at 600 epochs, 650 epochs, and 700 epochs respectively during the training process to save the model. We use Adam as the optimizer, it is an extension of stochastic gradient descent (SGD). It uses momentum and adaptive learning rate to speed up the convergence speed, which can replace the SGD to update the network weights more effectively. The deep learning framework used to build and train the model is Keras 2.1.3 using Tensorflow backend. The training is carried out using a PC with Intel Core i7-8700 CPUs and 8 GB RAM. A single training process takes about 2 h.

3.2. Performance of the model with different structures

Our proposed model Residual LSTM Plus is a modified neural network framework based on DRN and LSTM. We conduct a comparison to verify whether the performance of our proposed model is improved compared to these two individual models. Fig. 10 shows the training losses of these models, we can see that compared to DRN and LSTM, the training losses are significantly reduced, which proves that the Residual LSTM Plus framework can indeed greatly improve the performance of the model.

In Section 2.4, we introduce the dimension weighted unit used to learn the interdependencies between feature dimensions. Table 3 shows the influence of the use of the dimension weighted unit and the value of the scaling hyperparameter r that controls the capacity of the dimension weight unit and the calculation cost on the model performance. In theory, when r increases, the complexity of the model decreases, but the accuracy also decreases accordingly. From the results, we can see that when the hyperparameter r of the dimension weighted unit changes, a smaller r increases the model complexity, but the performance of the model does not monotonically increase with the increase in model complexity, and the change of r has no obvious influence on model performance. This is similar to the experimental results in [21], the reason is that the dimension weight unit overfits the interdependencies between the feature dimensions, making the model performance robust within a certain r interval.

[1] Available at https://www.iso-ne.com/isoexpress/web/reports/load-and-demand.

[2] Available at https://data.mendeley.com/datasets/f4fcrh4tn9/1.

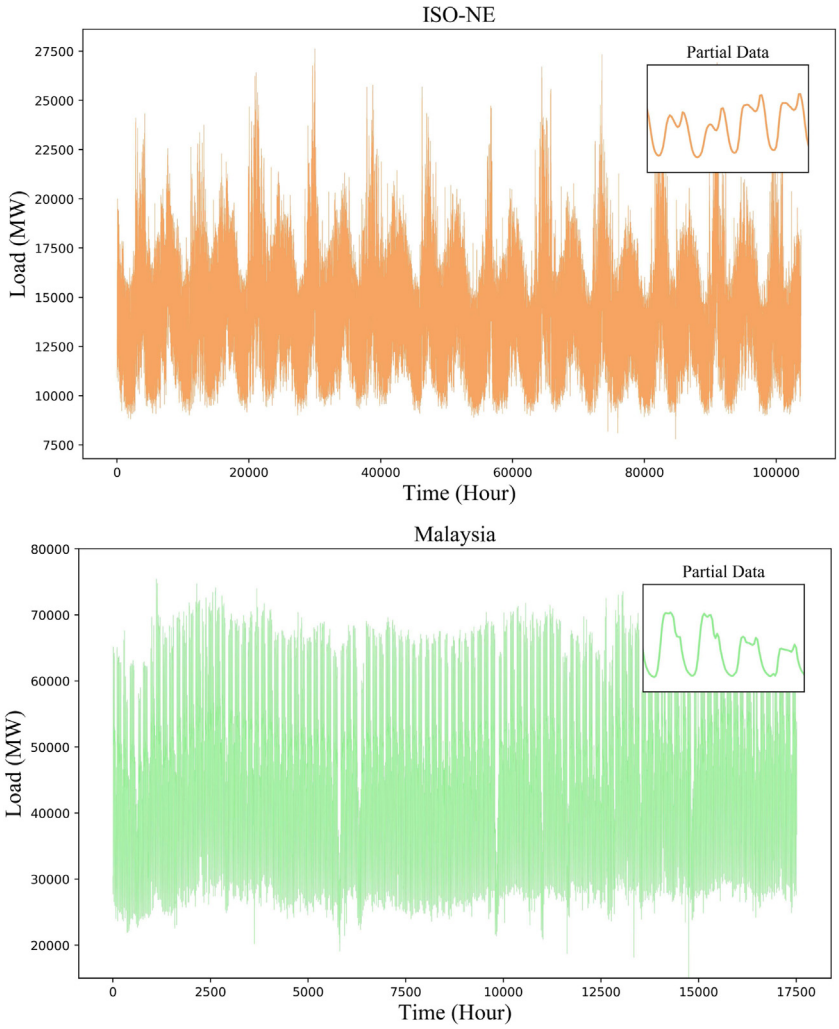


Fig. 9. Load data in the ISO-NE and Malaysia datasets.

Table 2
Hyperparameter settings of multiple mainstream neural network models involved in this paper.

Model	Dropout rate	Neurons	Optimizer	Batch size	Epoch	Snapshot
Fully-connected network	NULL	24	Adam	32	700	3
RNN	0.5	256	Adam	32	700	3
LSTM(w/o DWU)	0.5	256	Adam	32	700	3
LSTM(with DWU)	0.5	256	Adam	32	700	3
DRN	0.5	24	Adam	32	700	3
ResNet Plus	NULL	24	Adam	32	700	3
Proposed model	0.5	256	Adam	32	700	3

Table 3
MAPE of the proposed model using different hyperparameters r.

r	w/o	1	2	4	8	16	32	64	128	256
MAPE	1.73	1.60	1.60	1.60	1.61	**1.56**	1.65	1.59	1.62	1.65

Compared with the model without the dimension weighted unit (w/o represents the model without dimension weighted unit), the error is reduced regardless of the value of r, which proves the effectiveness of the dimension weighted unit. We can see that when $r = 16$, the model has the relatively best performance, and a good balance is achieved between accuracy and model complexity. Therefore, in the experimental part, we choose $r = 16$ as our default value.

Table 4
Performance of the model with/without specific methods.

Method	With	Without
Dropout	**1.56**	1.76
BN	**1.56**	1.67
Snapshot ensemble method	**1.56**	1.62
Proposed penalty term	**1.56**	1.99 (using MSE)

Table 4 shows the performance of the proposed model with/without specific methods including dropout, BN, ensemble method and penalty term. The experimental results prove that the application of these excellent methods and algorithms can indeed help improve the performance of our proposed model.

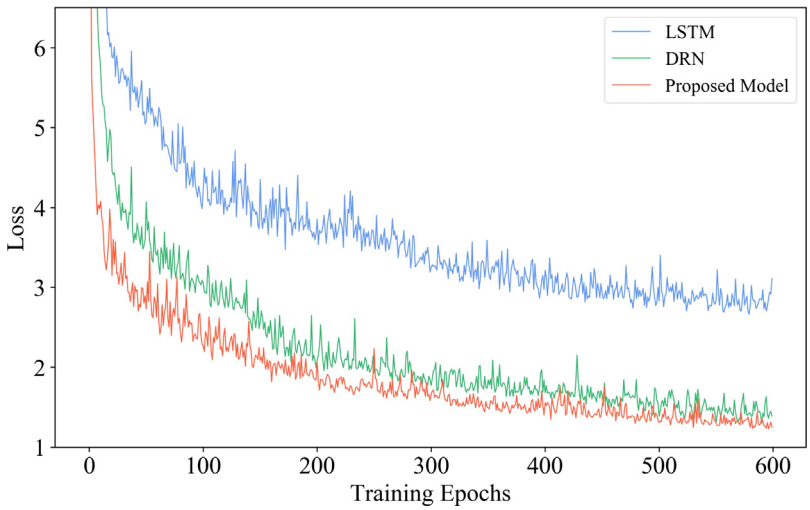


Fig. 10. Train losses of DRN, LSTM and our proposed model.

Table 5
MAPE of the model structures at different depths.

Model	Number of LSTM layers	Number of residual LSTM blocks	MAPE
Stacked LSTM(w/o DWU)	2	NULL	2.85
Stacked LSTM(with DWU)	2	NULL	2.63
Residual LSTM Plus	2	2	**1.72**
Stacked LSTM(w/o DWU)	4	NULL	2.57
Stacked LSTM(with DWU)	4	NULL	2.48
Residual LSTM Plus	4	4	**1.63**
Stacked LSTM(w/o DWU)	8	NULL	3.23
Stacked LSTM(with DWU)	8	NULL	2.85
Residual LSTM Plus	8	8	**1.61**
Stacked LSTM(w/o DWU)	16	NULL	4.32
Stacked LSTM(with DWU)	16	NULL	3.99
Residual LSTM Plus	16	16	**1.59**
Stacked LSTM(w/o DWU)	20	NULL	4.67
Stacked LSTM(with DWU)	20	NULL	4.37
Residual LSTM Plus	20	20	**1.56**

This is also an important process in the optimization and improvement of the proposed model structure.

We know that the residual network can help us significantly increase the depth of the neural network, a typical LSTM network model only contains 2–4 layers of LSTM. When the number of layers continues to increase, the vanishing/exploding gradient and degradation problem may greatly reduce the accuracy of the model. We then test whether our proposed Residual LSTM Plus combined with the residual network architecture can be trained to a very deep depth. Table 5 compares the performance of our proposed model with the typical LSTM at different depths. It can be seen that when the number of LSTM layers reaches 20, our model still has the best performance, the MAPE is only 1.56, which is one third of MAPE of stacked LSTM with the same number of layers. On the contrary, the accuracy of LSTM without residual network structure has significantly reduced when the number of layers increases to a certain extent. The results prove that the residual network architecture can indeed help us train deeper neural networks to obtain more accurate forecasts. From Table 5, we can also see that whether it is Residual LSTM Plus or stacked LSTM (with DWU), the MAPE of the network structure using the dimension weight unit is lower than that of the model without the dimension weight unit, and the accuracy has increased by up to 12%, which proves once again that the dimension weighted unit plays an important role in the proposed model. The above several experiments have indicated that by combining the LSTM with ResNet Plus and the dimension weight unit, we can jointly improve the performance of the proposed model from the three aspects of depth, time, and feature dimensions.

Table 6
Comparison of the proposed model with mainstream models in ISO-NE and Malaysia datasets.

Model	ISO-NE	Malaysia
Fully-connected neural network	2.10	5.09
RNN	2.83	5.10
LSTM(w/o DWU)	2.57	4.66
LSTM(with DWU)	2.48	4.29
DRN	1.80	4.61
ResNet Plus	1.75	4.50
Linear regression	14.04	11.18
SVM	13.29	11.28
ARIMA	1.85	4.82
Proposed model	**1.56**	**4.10**

3.3. Comparison with mainstream models

We compare the proposed Residual LSTM Plus model with multiple existing mainstream models in ISO-NE and Malaysia datasets. The experimental results are shown in Table 6. It can be seen that on the two datasets, our proposed model outperforms other existing models, which shows that our model has excellent load forecast accuracy and generalization capability. Moreover, compared with traditional machine learning methods (SVM) and classic parameterization methods (linear regression, ARIMA), modern learning methods based on neural networks have relatively better performance and generalization capabilities on load forecasting problems, which shows that they have great development potential and research value.

3.4. Performance of the proposed model in different periods

In the end, we test the performance of the proposed model in different years and seasons to further verify and compare the actual forecast accuracy and generalization capability of the model in different periods. LSTM and ResNet Plus are also used for comparison. From Table 7, we can see that the performance of our proposed model on the three-year test set is better than that of the LSTM and ResNet Plus models in most cases through

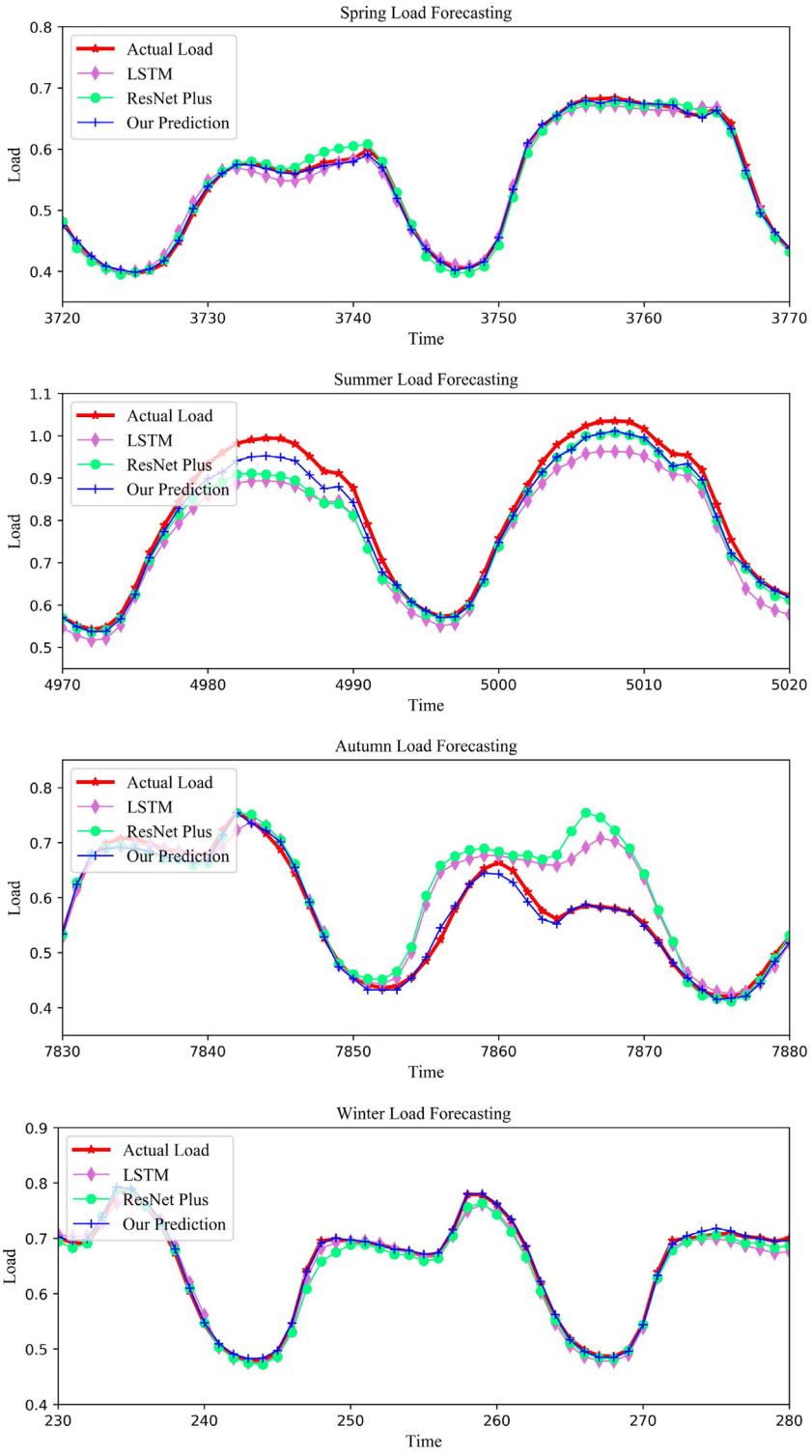


Fig. 11. Performance of the models in different seasons. (forecast values are normalized).

the coefficient of determination R^2. Compared with other models, the R^2 of the proposed model is closer to 1, which proves that the predicted value of the model can fit the actual observed value well. Moreover, when forecasting the load in different years, the results do not fluctuate too much, which indicates that the proposed model is also robust through the ensemble method.

Fig. 11 shows the comparison between these models' load forecast curves and the actual load curve, we can see that in the four seasons, the forecast curve of Residual LSTM Plus is the closest to the actual value, which shows that our proposed model can forecast the load accurately in different seasons. Fig. 12 also demonstrates that the proposed model can fit the actual load curve well both on weekday/weekend and holiday. The above experiments indicate that our proposed model has the most accurate forecast results in different periods, which proves the great reliability and generalization capability of it.

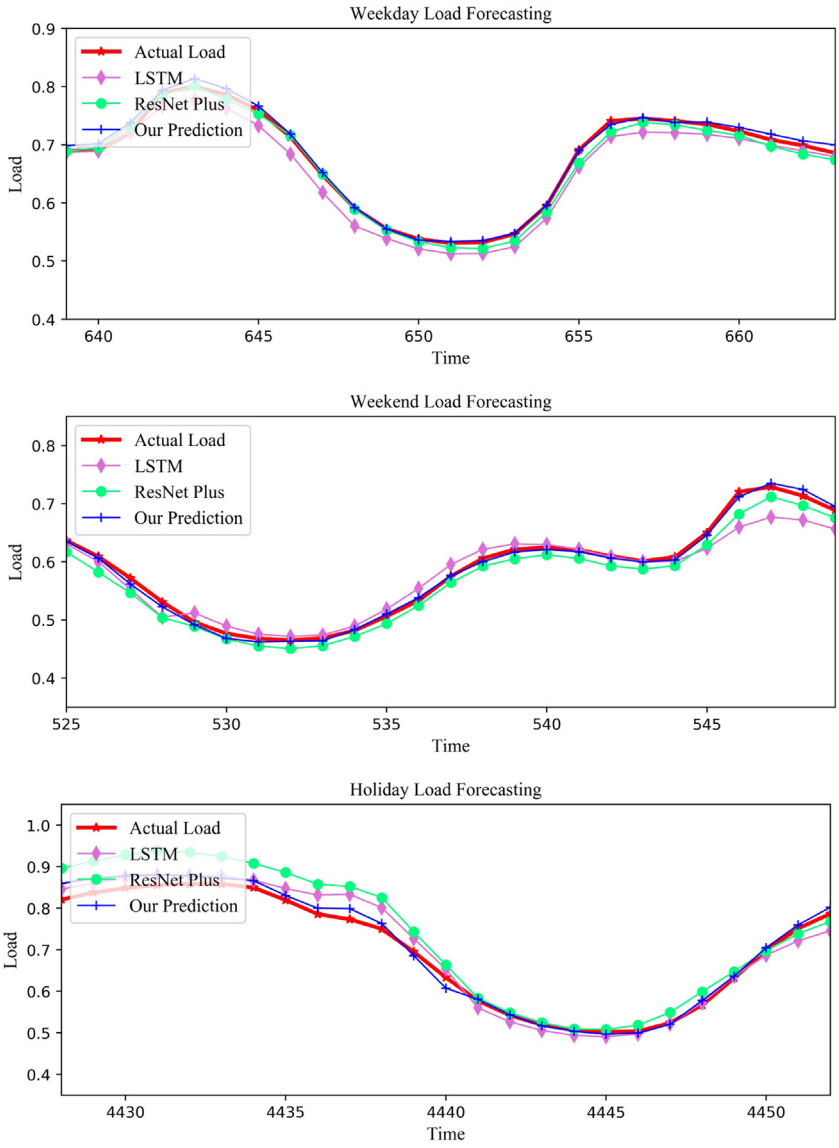


Fig. 12. Performance of the models on weekday/weekend and holiday. (forecast values are normalized).

Table 7
Performance of the models in different years.

Model	2008	2009	2010
LSTM	0.953	0.959	0.957
ResNet Plus	0.979	0.980	**0.980**
Proposed model	**0.980**	**0.982**	**0.980**

4. Conclusion

In this paper, we have proposed a neural network framework Residual LSTM Plus based on DRN and LSTM to perform STLF. This proposed model consists of a fully connected neural network and a modified residual network including LSTM and dimension weight units. The modified residual network structure ResNet Plus enhances the expressive ability of the model and the efficiency of back propagation through dense shortcut connections; LSTM enables the model to learn the hidden relationship of the input information in the time dimension; the dimension weight unit can explicitly model the interdependencies between feature dimensions. Combining these three advanced neural network structures, our proposed Residual LSTM Plus framework can significantly improve the performance of the model from the three aspects of depth, time, and feature dimension. The snapshot ensemble method is also applied to further improve the accuracy and robustness of the proposed model. Through multiple sets of controlled experiments, we verify that the proposed model has a significant performance improvement compared to using ResNet Plus and LSTM alone, which proves its effectiveness; we also compare Residual LSTM Plus with mainstream models, showing that the proposed model has state-of-the-art performance in performing STLF tasks; finally, we also test the model on different datasets and periods, proving that it has excellent generalization capability.

In future work, we hope to propose more excellent neural network structures and apply them into actual load forecasting

problems, so that we can further release the potential of neural networks to improve the performance of our STLF model.

CRediT authorship contribution statement

Ziyu Sheng: Conceptualization, Methodology, Software, Writing – original draft. **Zeyu An:** Validation, Formal analysis, Data curation. **Huiwei Wang:** Investigation, Project administration, Funding acquisition. **Guo Chen:** Supervision, Writing – review & editing. **Kun Tian:** Visualization, Resources.

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

I have shared the link to my data

Acknowledgments

This work was supported in part by the Natural Science Foundation of Chongqing, China under Grant cstc2020jcyj-msxmX0057, and in part by the Fundamental Research Funds for the Central Universities, China under Grant XDJK2018B013. The statements made herein are solely the responsibility of the authors.

References

[1] I. Koprinska, M. Rana, V.G. Agelidis, Correlation and instance based feature selection for electricity load forecasting, Knowl.-Based Syst. 82 (2015) 29–40.
[2] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, J. He, Short-term load forecasting with deep residual networks, IEEE Trans. Smart Grid 10 (4) (2018) 3943–3952.
[3] E. Feilat, M. Bouzguenda, Medium-term load forecasting using neural network approach, in: 2011 IEEE PES Conference on Innovative Smart Grid Technologies-Middle East, IEEE, 2011, pp. 1–5.
[4] G. Nalcaci, A. Özmen, G.W. Weber, Long-term load forecasting: models based on MARS, ANN and LR methods, CEJOR Cent. Eur. J. Oper. Res. 27 (4) (2019) 1033–1049.
[5] N. Amral, C. Ozveren, D. King, Short term load forecasting using multiple linear regression, in: 2007 42nd International Universities Power Engineering Conference, IEEE, 2007, pp. 1192–1198.
[6] A.-S. Nazih, E. Fawwaz, A. Osama M, Medium-term electric load forecasting using multivariable linear and non-linear regression, Smart Grid Renew. Energy 2 (2) (2011) 126–135.
[7] W. Xu, H. Peng, X. Zeng, F. Zhou, X. Tian, X. Peng, A hybrid modelling method for time series forecasting based on a linear regression model and deep learning, Appl. Intell. 49 (8) (2019) 3002–3015.
[8] D. Niu, Y. Wang, C. Duan, M. Xing, A new short-term power load forecasting model based on chaotic time series and SVM, J. UCS 15 (13) (2009) 2726–2745.
[9] Z. Junfang, W. Yiang, W. Junji, Application of gray system theory in load forecasting, Electr. Power Autom. Equip. 24 (5) (2004) 24–27.
[10] I.P. Panapakidis, Application of hybrid computational intelligence models in short-term bus load forecasting, Expert Syst. Appl. 54 (2016) 105–120.
[11] P.K.d.M.M. Freire, C.A.G. Santos, G.B.L. da Silva, Analysis of the use of discrete wavelet transforms coupled with ANN for short-term streamflow forecasting, Appl. Soft Comput. 80 (2019) 494–505.
[12] Z. Sheng, H. Wang, G. Chen, B. Zhou, J. Sun, Convolutional residual network to short-term load forecasting, Appl. Intell. 51 (4) (2021) 2485–2499.
[13] X. Yuan, C. Chen, M. Jiang, Y. Yuan, Prediction interval of wind power using parameter optimized beta distribution based LSTM model, Appl. Soft Comput. 82 (2019) 105550.
[14] J. Kim, M. Elkhamy, J. Lee, Residual LSTM: Design of a deep recurrent architecture for distant speech recognition, 2017, arXiv preprint arXiv:1701.03360.
[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[16] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[17] W. Kong, Z.Y. Dong, Y. Jia, D.J. Hill, Y. Xu, Y. Zhang, Short-term residential load forecasting based on LSTM recurrent neural network, IEEE Trans. Smart Grid 10 (1) (2017) 841–851.
[18] A. Prakash, S.A. Hasan, K. Lee, V.V. Datla, A. Qadir, J. Liu, O. Farri, Neural paraphrase generation with stacked residual LSTM networks, 2016, arXiv preprint arXiv:1610.03098.
[19] X. Tong, C.-W. Huang, S.H. Mallidi, S. Joseph, S. Pareek, C. Chandak, A. Rastrow, R. Maas, Streaming ResLSTM with causal mean aggregation for device-directed utterance detection, 2020, arXiv preprint arXiv:2007.09245.
[20] Y. Li, D. Tang, G. Jiang, X. Zhitao, L. Geng, F. Zhang, J. Wu, Short term traffic flow forecasting based on dimension weighted residual LSTM, Comput. Eng. 45 (6) (2019) 1–5.
[21] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.
[22] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012, arXiv preprint arXiv:1207.0580.
[23] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, arXiv preprint arXiv:1502.03167.
[24] G. Huang, Y. Li, G. Pleiss, Z. Liu, J.E. Hopcroft, K.Q. Weinberger, Snapshot ensembles: Train 1, get m for free, 2017, arXiv preprint arXiv:1704.00109.
[25] J. Moon, S. Park, S. Rho, E. Hwang, A comparative analysis of artificial neural network architectures for building energy consumption forecasting, Int. J. Distrib. Sens. Netw. 15 (9) (2019) 1550147719877616.
[26] S.K. Acharya, Y.-M. Wi, J. Lee, Short-term load forecasting for a single household based on convolution neural networks using data augmentation, Energies 12 (18) (2019) 3560.
[27] N. Liu, Q. Tang, J. Zhang, W. Fan, J. Liu, A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids, Appl. Energy 129 (2014) 336–345.
[28] D. Duvenaud, O. Rippel, R. Adams, Z. Ghahramani, Avoiding pathologies in very deep networks, in: Artificial Intelligence and Statistics Conference, 2014, pp. 202–210.
[29] A.E. Orhan, X. Pitkow, Skip connections eliminate singularities, 2017, arXiv preprint arXiv:1701.09175.
[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[31] J. Pennington, S. Schoenholz, S. Ganguli, Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice, in: Advances in Neural Information Processing Systems, 2017, pp. 4785–4795.
[32] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
[33] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, J. Feng, Dual path networks, in: Advances in Neural Information Processing Systems, 2017, pp. 4467–4475.
[34] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.
[35] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al., Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.
[36] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, Y. Gao, Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 631–648.
[37] T. Strauss, M. Hanselmann, A. Junginger, H. Ulmer, Ensemble methods as a defense to adversarial perturbations against deep neural networks, 2018, arXiv preprint arXiv:1709.03423.
[38] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in: Advances in Neural Information Processing Systems, 2017, pp. 971–980.
[39] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[40] H.J. Sadaei, P.C.d.L. e Silva, F.G. Guimarães, M.H. Lee, Short-term load forecasting by using a combined method of convolutional neural networks and fuzzy time series, Energy 175 (2019) 365–377.