Short-Term Load Forecasting With Deep Residual Networks

Kunjin Chen, Kunlong Chen, Qin Wang, Ziyu He, Jun Hu, *Member, IEEE*, and Jinliang He, *Fellow, IEEE*

***Abstract*—We present in this paper a model for forecasting short-term electric load based on deep residual networks. The proposed model is able to integrate domain knowledge and researchers' understanding of the task by virtue of different neural network building blocks. Specifically, a modified deep residual network is formulated to improve the forecast results. Further, a two-stage ensemble strategy is used to enhance the generalization capability of the proposed model. We also apply the proposed model to probabilistic load forecasting using Monte Carlo dropout. Three public datasets are used to prove the effectiveness of the proposed model. Multiple test cases and comparison with existing models show that the proposed model provides accurate load forecasting results and has high generalization capability.**

***Index Terms*—Short-term load forecasting, deep learning, deep residual network, probabilistic load forecasting.**

I. Introduction

THE FORECASTING of power demand is of crucial importance for the development of modern power systems. The stable and efficient management, scheduling and dispatch in power systems rely heavily on precise forecasting of future loads on various time horizons. In particular, short-term load forecasting (STLF) focuses on the forecasting of loads from several minutes up to one week into the future [1]. A reliable STLF helps utilities and energy providers deal with the challenges posed by the higher penetration of renewable energies and the development of electricity markets with increasingly complex pricing strategies in future smart grids.

Various STLF methods have been proposed by researchers over the years. Some of the models used for STLF include linear or nonparametric regression [2], [3], support vector regression (SVR) [1], [4], autoregressive models [5], fuzzy-logic approach [6], etc. Reviews and evaluations of existing methods can be found in [7]–[10]. Building STLF systems with artificial neural networks (ANN) has long been one of the main-stream solutions to this task. As early as 2001, a review paper by Hippert *et al*. surveyed and examined a collection of papers that had been published between 1991 and 1999, and arrived at the conclusions that most of the proposed models were over-parameterized and the results they had to offer were not convincing enough [11]. In addition to the fact that the size of neural networks would grow rapidly with the increase in the numbers of input variables, hidden nodes or hidden layers, other criticisms mainly focus on the "overfitting" issue of neural networks [1]. Nevertheless, different types and variants of neural networks have been proposed and applied to STLF, such as radial basis function (RBF) neural networks [12], wavelet neural networks [13], [14], extreme learning machines (ELM) [15], to name a few.

Recent developments in neural networks, especially deep neural networks, have had great impacts in the fields including computer vision, natural language processing, and speech recognition [16]. Instead of sticking with fixed shallow structures of neural networks with hand-designed features as inputs, researchers are now able to integrate their understandings of different tasks into the network structures. Different building blocks including convolutional neural networks (CNN) [17], and long short-term memory (LSTM) [18] have allowed deep neural networks to be highly flexible and effective. Various techniques have also been proposed so that neural networks with many layers can be trained effectively without the vanishing of gradients or severe overfitting. Applying deep neural networks to short-term load forecasting is a relatively new topic. Researchers have been using restricted Boltzmann machines (RBM) and feed-forward neural networks with multiple layers in forecasting of demand side loads and natural gas loads [19], [20]. However, these models are increasingly hard to train as the number of layers increases, thus the number of hidden layers are often considerably small (e.g., 2 to 5 layers), which limits the performance of the models.

In this work, we aim at extending existing structures of ANN for STLF by adopting state-of-the-art deep neural network structures and implementation techniques. Instead of stacking multiple hidden layers between the input and the output, we learn from the residual network structure proposed in [21] and propose a novel end-to-end neural network model capable of forecasting loads of next 24 hours. An ensemble strategy to combine multiple individual networks is also proposed. Further, we extend the model to probabilistic load forecasting by adopting Monte Carlo (MC) dropout (for a

Manuscript received October 1, 2017; revised February 1, 2018 and March 18, 2018; accepted May 29, 2018. Date of publication June 5, 2018; date of current version June 19, 2019. Paper no. TSG-01422-2017. *(Corresponding author: Jinliang He.)*

K. Chen, J. Hu, and J. He are with the State Key Laboratory of Power Systems, Department of Electrical Engineering, Tsinghua University, Beijing 100084, China (e-mail: hejl@tsinghua.edu.cn).

K. Chen is with the Department of Electrical Engineering, Beijing Jiaotong University, Beijing 100044, China.

Q. Wang is with the Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zürich, Switzerland.

Z. He is with the Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90007 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSG.2018.2844307

comprehensive review of probabilistic electric load forecasting, the reader is referred to [22] and [23]). The contributions of this work are two-folds. First, a fully end-to-end model based on deep residual networks for STLF is proposed. The proposed model does not involve external feature extraction or feature selection algorithms, and only raw data of loads, temperature and information that is readily available are used as inputs. The results show that the forecasting performance can be greatly enhanced by improving the structure of the neural networks and adopting the ensemble strategy. As complicated feature engineering techniques and additional information (e.g., humidity, wind speed, cloud cover, etc.) are not involved, we provide a good benchmark that can be easily compared with. In addition, the building blocks of the proposed model can also be adapted to existing neural-network-based STLF models. Combining the building blocks with existing feature extraction and feature selection techniques is straight-forward and may lead to further improvement in accuracy. Additional data can also be easily incorporated. Second, a new formulation of probabilistic STLF for an ensemble of neural networks is proposed. By using MC dropout, we can directly obtain the probability forecasting results using the models trained for the task of point forecasting.

The remainder of the paper is organized as follows. In Section II, we formulate the proposed model based on deep residual networks. The ensemble strategy, the MC dropout method, as well as the implementation details are also provided. In Section III, the results of STLF by the proposed model are presented. We also discuss the performance of the proposed model and compare it with existing methods. Section IV concludes this paper and proposes future works. The source code for the STLF model proposed in this paper is available at https://github.com/yalickj/load-forecasting-resnet.

II. Short-Term Load Forecasting Based on Deep Residual Networks

In this paper, we propose a day-ahead load forecasting model based on deep residual networks. We first formulate the low-level basic structure where the inputs of the model are processed by several fully connected layers to produce preliminary forecasts of 24 hours. The preliminary forecasts are then passed through a deep residual network. After presenting the structure of the deep residual network, some modifications are made to further enhance its learning capability. An ensemble strategy is designed to enhance the generalization capability of the proposed model. The formulation of MC dropout for probabilistic forecasting is also provided.

A. Model Input and the Basic Structure for Load Forecasting of One Hour

We use the model with the basic structure to give preliminary forecasts of the 24 hours of the next day. Specifically, the inputs used to forecast the load for the hth hour of the next day, L_h, are listed in Table I. The values for loads and temperatures are normalized by dividing the maximum value of the training dataset. The selected inputs allow us to capture both short-term closeness and long-term trends in the load and temperature time series [24]. More specifically, we expect that L_h^{month}, L_h^{week}, T_h^{month} and T_h^{week} can help the model identify long-term trends in the time series (the days of the same day-of-week index as the next day are selected as they are more likely to have similar load characteristics [13]), while L_h^{day} and T_h^{day} are able to provide short-term closeness and characteristics. The input L_h^{hour} feeds the loads of the most recent 24 hours to the model. Forecast loads are used to replace the values in L_h^{hour} that are not available at the time of forecasting, which also helps associate the forecasts of the whole day. Note that the sizes of the above-mentioned inputs can be adjusted flexibly. In addition, one-hot codes for season,[1] weekday/weekend distinction, and holiday/non-holiday[2] distinction are added to help the model capture the periodic and unordinary temporal characteristics of the load time series.

TABLE I
Inputs for the Load Forecast of the hth Hour of the Next Day

Input	Size	Description of the Inputs
L_h^{month}	6	Loads of the hth hour of the days that are 4, 8, 12, 16, 20, and 24 weeks prior to the next day
L_h^{week}	4	Loads of the hth hour of the days that are 1, 2, 3, and 4 weeks prior to the next day
L_h^{day}	7	Loads of the hth hour of every day of the week prior to the next day
L_h^{hour}	24	Loads of the most recent 24 hours prior to the hth hour of the next day
T_h^{month}	6	Temperature values of the same hours as L_h^{month}
T_h^{week}	4	Temperature values of the same hours as L_h^{week}
T_h^{day}	7	Temperature values of the same hours as L_h^{day}
T_h	1	The actual temperature of the hth hour of the next day
S	4	One-hot code for season
W	2	One-hot code for weekday/weekend distinction
H	2	One-hot code for holiday/non-holiday distinction

The structure of the neural network model for load forecasting of one hour is illustrated in Fig. 1. For L_h^{month}, L_h^{week}, L_h^{day}, T_h^{month}, T_h^{week}, and T_h^{day}, we first concatenate the pairs $[L_h^{month}, T_h^{month}]$, $[L_h^{week}, T_h^{week}]$, and $[L_h^{day}, T_h^{day}]$, and connect them with three separate fully-connected layers. The three fully-connected layers are then concatenated and connected with another fully-connected layer denoted as FC_2. For L_h^{hour}, we forward pass it through two fully-connected layers, the second layer of which is denoted as FC_1. S and W are concatenated to produce two fully-connected layers, one used as part of the input of FC_1, the other used as part of the input of FC_2. H is also connected to FC_2. In order to produce the output L_h, we concatenate FC_1, FC_2, and T_h, and connect them with a fully-connected layer, which is then connected to L_h with another fully connected layer. All fully-connected layers

[1]In this paper, the ranges for Spring, Summer, Autumn, and Winter are March 8th to June 7th, June 8th to September 7th, September 8th to December 7th, December 8th to March 7th, respectively.

[2]In this paper, we consider three major public holidays, namely, Christmas Eve, Thanksgiving Day, and Independence Day as the activities involved in these holidays have great impacts on the loads. The rest of the holidays are considered as non-holidays for simplicity.

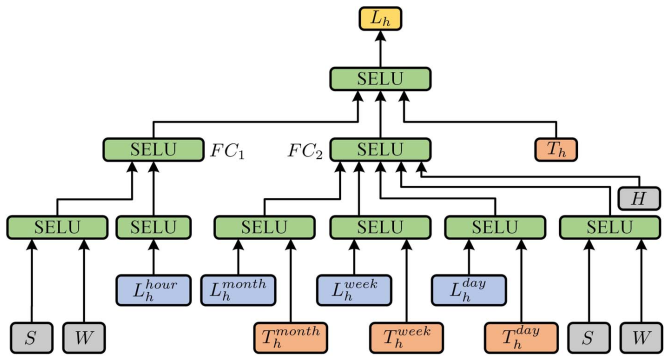


Fig. 1. The structure of the neural network model for load forecasting of one hour.

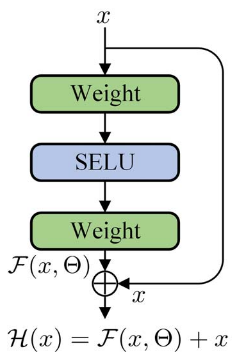


Fig. 2. The building block of the deep residual network. SELU is used as the activation function between two linear layers.

but the output layer use scaled exponential linear units (SELU) as the activation function.

The adoption of the ReLU has greatly improved the performance of deep neural networks [25]. Specifically, ReLU has the form

$$\mathrm{ReLU}(y_i) = \max(0, y_i) \tag{1}$$

where y_i is the linear activation of the i-th node of a layer. A problem with ReLU is that if a unit can not be activated by any input in the dataset, the gradient-based optimization algorithm is unable to update the weights of the unit, so that the unit will never be activated again. In addition, the network will become very hard to train if a large proportion of the hidden units produce constant 0 gradients [26]. This problem can be solved by adding a slope to the negative half axis of ReLU. With a simple modification to the formulation of ReLU on the negative half axis, we get PReLU [27]. The activations of a layer with PReLU as the activation function is obtained by

$$\mathrm{PReLU}(y_i) = \begin{cases} y_i & \text{if } y_i > 0 \\ \beta_i y_i & \text{if } y_i \le 0 \end{cases} \tag{2}$$

where β_i is the coefficient controlling the slope of $\beta_i y_i$ when $y_i \le 0$. A further modification to ReLU that induces self-normalizing properties is provided in [28], where the activation function of SELU is given by

$$\mathrm{SELU}(y_i) = \lambda \begin{cases} y_i & \text{if } y_i > 0 \\ \alpha e^{y_i} - \alpha & \text{if } y_i \le 0 \end{cases} \tag{3}$$

where λ and α are two tunable parameters. It is shown in [28] that if we have $\lambda \approx 1.0577$ and $\alpha \approx 1.6733$, the outputs of the layers in a fully-connected neural network would approach the standard normal distribution when the inputs follow the standard normal distribution. This helps the networks to prevent the problems of vanishing and exploding gradients.

As previously mentioned, in order to associate the forecasts of the 24 hours of the next day, the corresponding values within L_h^{hour} are replaced by $\{L_1, \ldots, L_{h-1}\}$ for $h > 1$. Instead of simply copying the values, we maintain the neural network connections underneath them. Thus, the gradients of subsequent hours can be propagated backward through time. This would help the model adjust the forecast value of each hour given the inputs and forecast values of the rest of the hours.

We then concatenate $\{L_1, \ldots, L_{24}\}$ as L, which directly becomes the output of the model with the basic structure. Next, we proceed to formulate the deep residual network and add it on top of L. The output of the deep residual network is denoted as $\hat{y}$ and has the same size of L.

B. The Deep Residual Network Structure for Day-Ahead Load Forecasting

In [21], an innovative way of constructing deep neural networks for image recognition is proposed. In this paper, the residual block in Fig. 2 is used to build the deep neural network structure. In the residual block, instead of learning a mapping from x to $\mathcal{H}(x)$, a mapping from x to $\mathcal{F}(x, \Theta)$ is learned, where Θ is a set of weights (and biases) associated with the residual block. Thus, the overall representation of the residual block becomes

$$\mathcal{H}(x) = \mathcal{F}(x, \Theta) + x \tag{4}$$

A deep residual network can be easily constructed by stacking a number of residual blocks. We illustrate in Fig. 3 the structure of the deep residual network (ResNet) used for the proposed model. More specifically, if K residual blocks are stacked, the forward propagation of such a structure can be represented by

$$x_K = x_0 + \sum_{i=1}^{K} \mathcal{F}(x_{i-1}, \Theta_{i-1}) \tag{5}$$

where x_0 is the input of the residual network, x_K the output of the residual network, and $\Theta_i = \{\theta_{i,l}|_{1 \le l \le L}\}$ the set of weights associated with the ith residual block, L being the number of layers within the block. The back propagation of the overall loss of the neural network to x_0 can then be calculated as

$$\frac{\partial \mathcal{L}}{\partial x_0} = \frac{\partial \mathcal{L}}{\partial x_K}\left(1 + \frac{\partial}{\partial x_0}\sum_{i=1}^{K} \mathcal{F}(x_{i-1}, \Theta_{i-1})\right) \tag{6}$$

where $\mathcal{L}$ is the overall loss of the neural network. The "1" in the equation indicates that the gradients at the output of the network can be directly back-propagated to the input of the network, so that the vanishing of gradients (which is often observed when the gradients at the output have to go through many layers before reaching the input) in the network is much

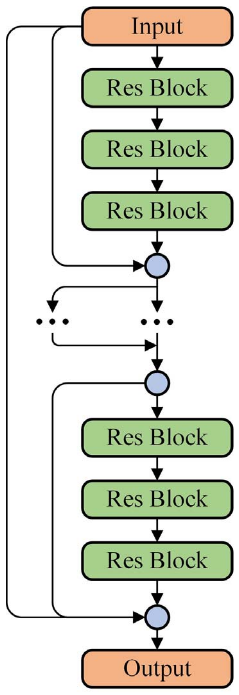


Fig. 3. An illustration of the deep residual network (ResNet) structure. More shortcut connections are made in addition to the ones within the blocks. In this figure, every three residual blocks has one shortcut connection and another shortcut connection is made from the input to the output. Each round node averages all of its inputs.

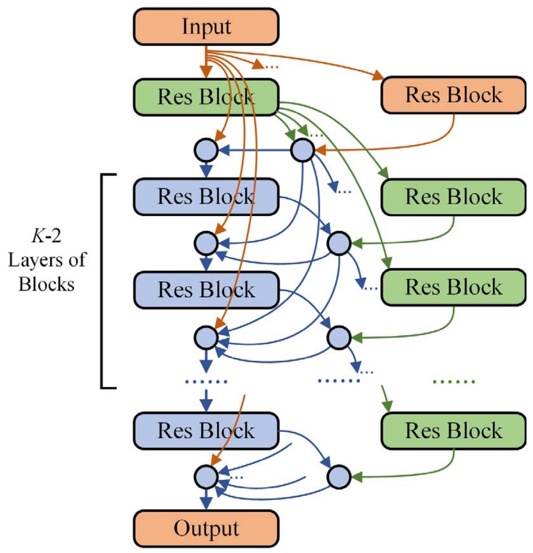


Fig. 4. An illustration of the modified deep residual network (ResNetPlus) structure. The blue dots in the figure average their inputs, and the outputs are connected to subsequent residual blocks.

less likely to occur [29]. As a matter of fact, this equation can also be applied to any pair (x_i, x_j) $(0 \leq i < j \leq K)$, where x_i and x_j are the output of the ith residual block (or the input of the network when $i = 0$), and the jth residual block, respectively.

In addition to the stacked residual blocks, extra shortcut connections can be added into the deep residual network, as is introduced in [30]. Concretely, two levels of extra shortcut connections are added to the network. The lower level shortcut connection bypasses several adjacent residual blocks, while the higher level shortcut connection is made between the input and output. If more than one shortcut connection reaches a residual block or the output of the network, the values from the connections are averaged. Note that after adding the extra shortcut connections, the formulations of the forward-propagation of responses and the back-propagation of gradients are slightly different, but the characteristics of the network that we care about remain unchanged.

We can further improve the learning ability of ResNet by modifying its structure. Inspired by the convolutional network structures proposed in [31] and [32], we propose the modified deep residual network (ResNetPlus), whose structure is shown in Fig. 4. First, we add a series of side residual blocks to the model (the residual blocks on the right). Unlike the implementation in [32], the input of the side residual blocks is the output of the first residual block on the main path (except for the first side residual block, whose input is the input of the network). The output of each main residual block is averaged with the output of the side residual block in the same layer (indicated by the blue dots on the right). Similar to the densely connected network in [31], the outputs of those blue dots are connected to all main residual blocks in subsequent layers. Starting from the second layer, the input of each main residual block is obtained by averaging all connections from the blue dots on the right together with the connection from the input of the network (indicated by the blue dots on the main path). It is expected that the additional side residual blocks and the dense shortcut connections can improve the representation capability and the efficiency of error back-propagation of the network. Later in this paper, we will compare the performance of the basic structure, the basic structure connected with ResNet, and the basic structure connected with ResNetPlus.

C. The Ensemble Strategy of Multiple Models

It is widely acknowledged in the field of machine learning that an ensemble of multiple models has higher generalization capability [16] than individual models. In [33], analysis of neural network ensembles for STLF of office buildings is provided by the authors. Results show that an ensemble of neural networks reduces the variance of performances. A demonstration of the ensemble strategy used in this paper is shown in Fig. 5. More specifically, the ensemble strategy consists of two stages.

The first stage of the strategy takes several snapshots during the training of a single model. Huang *et al.* [34] show that setting cyclic learning rate schedules for stochastic gradient descent (SGD) optimizer greatly improves the performance of existing deep neural network models. In this paper, as we use Adam (abbreviated from adaptive moment estimation [35]) as the optimizer, the learning rates for each iteration are decided adaptively. Thus, no learning rate schedules are set by ourselves. This scheme is similar to the NoCycle snapshot ensemble method discussed in [34], that is, we take several snapshots of the same model during its training process (e.g., the 4 snapshots along the training process of the model with

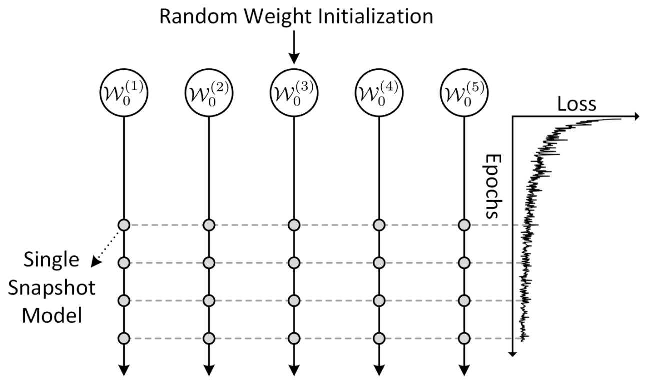


Fig. 5. A demonstration of the ensemble strategy used in this paper. The snapshot models are taken where the slope of validation loss is considerably small.

initial parameters $W_0^{(1)}$). As is indicated in Fig. 5, the snapshots are taken after an appropriate number of epochs, so that the loss of each snapshot is of similar level.

We can further ensemble a number of models that are trained independently. This is done by simply re-initializing the parameters of the model (e.g., $\mathcal{W}_0^{(1)}$ to $\mathcal{W}_0^{(5)}$ are 5 sets of initial parameters sampled from the same distribution used for initializing the model), which is one of the standard practices of obtaining good ensemble models [36]. The numbers of snapshots and re-trained models are hyper-parameters, which means they can be tuned using the validation dataset. After we obtain the all the snapshot models, we average the outputs of the models and produce the final forecast.

D. Probabilistic Forecasting Based on Monte Carlo Dropout

If we look at the deep residual network (either ResNet or ResNetPlus) as an ensemble of relatively shallow networks, the increased width and number of connections in the network can provide more shallow networks to form the ensemble model [32]. It is expected that the relatively shallow networks themselves can partially capture the nature of the load forecasting task, and multiple shallow networks with the same input can give varied outputs. This indicates that the proposed model have the potential to be used for probabilistic load forecasting.

Probabilistic forecasting of time series can be fulfilled by capturing the uncertainty within the models [37]. From a Bayesian probability theory point of view, the predictive probability of a Bayesian neural network can be obtained with

$$p\left(y^*|x^*\right) = \int_{\mathcal{W}} p\left(y^*|f^{\mathcal{W}}\left(x^*\right)\right) p\left(\mathcal{W}|X, Y\right) \mathrm{d}\mathcal{W} \tag{7}$$

where X and Y are the observations we use to train $f^{\mathcal{W}}(\cdot)$, a neural network with parameters $\mathcal{W}$. The intractable posterior distribution $p(\mathcal{W}|X, Y)$ is often approximated by various inference methods [37]. In this paper, we use MC dropout [38] to obtain the probabilistic forecasting uncertainty, which is easy and computationally efficient to implement. Specifically, dropout refers to the technique of randomly dropping out hidden units in a neural network during the training of the network [39], and a parameter p is used to control the probability that any hidden neuron is dropped out. If we apply dropout stochastically for M times at test time and collect the outputs of the network, we can approximate the first term of the forecasting uncertainty, which is

$$\begin{aligned}\mathrm{Var}\left(y^*|x^*\right) &= \mathrm{Var}\left[\mathbb{E}\left(y^*|\mathcal{W}, x^*\right)\right] + \mathbb{E}\left[\mathrm{Var}\left(y^*|\mathcal{W}, x^*\right)\right] \\ &= \mathrm{Var}\left(f^{\mathcal{W}}\left(x^*\right)\right) + \sigma^2 \\ &\approx \frac{1}{M}\sum_{m=1}^{M}\left(\hat{y}^*_{(m)} - \bar{\hat{y}}^*\right)^2 + \sigma^2 \end{aligned} \tag{8}$$

where $\hat{y}^*_{(m)}$ is the mth output we obtain, $\bar{\hat{y}}^*$ is the mean of all M outputs, and $\mathbb{E}$ denotes the expectation operator. The second term, σ^2, measures the *inherent noise* for the data generating process. According to [37], σ^2 can be estimated using an independent validation dataset. We denote the validation dataset with $X' = \{x'_1, \ldots, x'_V\}$, $Y' = \{y'_1, \ldots, y'_V\}$, and estimate σ^2 by

$$\sigma^2 = \frac{\beta}{V}\sum_{v=1}^{V}\left(y'_v - f^{\hat{\mathcal{W}}}\left(x'_v\right)\right)^2 \tag{9}$$

where $f^{\hat{\mathcal{W}}}(\cdot)$ is the model trained on the training dataset and β is a parameter to be estimated also using the validation dataset.

We need to extend the above estimation procedure to an ensemble of models. Concretely, for an ensemble of K neural network models of the same structure, we estimate the first term of (8) with a single model of the same structure trained with dropout. The parameter β in (9) is also estimated by the model. More specifically, we find the β that provides the best 90% and 95% interval forecasts on the validation dataset. σ^2 is estimated by replacing $f^{\hat{\mathcal{W}}}(\cdot)$ in (9) by the ensemble model, $f^*(\cdot)$. Note that the estimation of σ^2 is specific to each hour of the day.

After obtaining the forecasting uncertainty for each forecast, we can calculate the α-level interval with the point forecast, $f^*(x^*)$, and its corresponding quantiles to obtain probabilistic forecasting results.

E. Model Design and Implementation Details

The proposed model consists of the neural network structure for load forecasting of one hour (referred to as the basic structure), the deep residual network (referred to as ResNet) for improving the forecasts of 24 hours, and the modified deep residual network (referred to as ResNetPlus). The configurations of the models are elaborated as follows.

1) The Model With the Basic Structure: The graphic representation of the model with the basic structure is shown in Fig. 1. Each fully-connected layer for $[L_h^{day}, T_h^{day}]$, $[L_h^{week}, T_h^{week}]$, $[L_h^{month}, T_h^{month}]$, and L_h^{hour} has 10 hidden nodes, while the fully-connected layers for $[S, W]$ have 5 hidden nodes. FC_1, FC_2, and the fully-connected layer before L_h have 10 hidden nodes. All but the output layer use SELU as the activation function.

2) The Deep Residual Network (ResNet): ResNet is added to the neural network with the basic structure. Each residual block has a hidden layer with 20 hidden nodes and SELU as the activation function. The size of the outputs of the blocks is 24, which is the same as that of the inputs. A total of 30 residual blocks are stacked, forming a 60-layer deep residual network. The second level of shortcut connections is made every 5 residual blocks. The shortcut path of the highest level connects the input and the output of the network.

3) The Modified Deep Residual Network (ResNetPlus): The structure of ResNetPlus follows the structure shown in Fig. 4. The hyper-parameters inside the residual blocks are the same as ResNet.

In order to properly train the models, the loss of the model, $\mathcal{L}$, is formulated as the sum of two terms:

$$\mathcal{L} = \mathcal{L}_E + \mathcal{L}_R \tag{10}$$

where $\mathcal{L}_E$ measures the error of the forecasts, and $\mathcal{L}_R$ is an out-of-range penalty term used to accelerate the training process. Specifically, $\mathcal{L}_E$ is defined as

$$\mathcal{L}_E = \frac{1}{NH}\sum_{i=1}^{N}\sum_{h=1}^{H}\frac{\left|\hat{y}_{(i,h)} - y_{(i,h)}\right|}{y_{(i,h)}} \tag{11}$$

where $\hat{y}_{(i,h)}$ and $y_{(i,h)}$ are the output of the model and the actual normalized load for the hth hour of the ith day, respectively, N the number of data samples, and H the number of hourly loads within a day (i.e., $H = 24$ in this case). This error measure, widely known as the mean absolute percentage error (MAPE), is also used to evaluate the forecast results of the models. The second term, $\mathcal{L}_R$, is calculated as

$$\mathcal{L}_R = \frac{1}{2N}\sum_{i=1}^{N}\max\left(0, \max_h \hat{y}_{(i,h)} - \max_h y_{(i,h)}\right) + \max\left(0, \min_h y_{(i,h)} - \min_h \hat{y}_{(i,h)}\right) \tag{12}$$

This term penalizes the model when the forecast daily load curves are out of the range of the actual load curves, thus accelerating the beginning stage of the training process. When a model is able to produce forecasts with relatively high accuracy, this term serves to emphasize the cost for overestimating the peaks and the valleys of the load curves.

All the models are trained using the Adam optimizer with default parameters as suggested in [35]. The models are implemented using `Keras` 2.0.2 with `Tensorflow` 1.0.1 as backend in the Python 3.5 environment [40], [41]. A laptop with Intel Core i7-5500U CPUs is used to train the models. Training the ResNetPlus model with data of three years for 700 epochs takes approximately 1.5 hours. When 5 individual models are trained, the total training time is less than 8 hours.

III. Results and Discussion

In this section, we use the North-American Utility dataset[3] and the ISO-NE dataset[4] to verify the effectiveness of the proposed model. As we use actual temperature as the input, we further modify the temperature values to evaluate the performance of the proposed model. Results of probabilistic forecasting on the North-American Utility dataset and the GEFCom2014 dataset [42] are also provided.

[3]Available at https://class.ee.washington.edu/555/el-sharkawi.

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

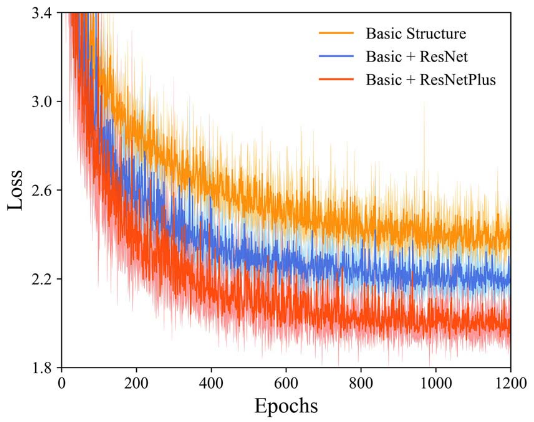


Fig. 6. Test losses of the neural network with the basic structure (Basic), the model with the deep residual network (Basic + ResNet), and the model with the modified deep residual network (Basic + ResNetPlus). Each model is trained 5 times with shuffled weight initialization. The solid lines are the average losses, and the standard deviation above and below the average losses are indicated by coloured areas.

A. Performance of the Proposed Model on the North-American Utility Dataset

The first test case uses the North-American Utility dataset. This dataset contains load and temperature data at one-hour resolution for a north-American utility. The dataset covers the time range between January 1st, 1985 and October 12th, 1992. The data of the two-year period prior to October 12th, 1992 is used as the test set, and the data prior to the test set is used for training the model. More specifically, two starting dates, namely, January 1st, 1986, and January 1st, 1988, are used for the training sets. As the latter starting date is used in experiments in the literature, we tune the hyper-parameters using the last 10% of the training set with this starting date.[5] The model trained with the training set containing 2 years of extra data has the same hyper-parameters.

Before reporting the performance of the ensemble model obtained by combining multiple individual models, we first look at the performance of the three models mentioned in Section II. The test losses of the three models are shown in Fig. 6 (the models are trained with the training set starting with January 1st, 1988). In order to yield credible results, we train each model 5 times and average the losses to obtain the solid lines in the figure. The coloured areas indicate the range between one standard deviation above and below the average losses. It is observed in the figure that ResNet is able to improve the performance of the model, and further reduction in loss can be achieved when ResNetPlus is implemented. Note that the results to be reported in this paper are all obtained with the ensemble model. For simplicity, the ensemble model with the basic structure connected with ResNetPlus is referred to as "the ResNetPlus model" hereinafter.

[5]For this dataset, 4 snapshots are taken between 1200 to 1350 epochs for 8 individual models. For the basic structure, all layers except the input and the output layers are shared for the 24 hours (sharing weights for 24 hours is only implemented in this test case). The ResNetPlus model has 30 layers on the main path.

TABLE II
COMPARISON OF THE PROPOSED RESNETPLUS MODEL WITH EXISTING MODELS ON THE NORTH-AMERICAN UTILITY DATASET WITH RESPECT TO MAPE (%)

Model	Actual temperature	Modified temperature
WT-NN [43]	2.64	2.84
WT-NN [44]	2.04	-
ESN [45]	2.37	2.53
SSA-SVR [1]	1.99	2.03
WT-ELM-MABC [46]	1.87	1.95
CLPSO-MA-SVR [47]	1.80	1.85
WT-ELM-LM [48]	1.67	1.73
Proposed model	**1.665**	**1.693**
Proposed model (2 extra years)	**1.557**	**1.575**

We compare the results of the proposed ResNetPlus model with existing models proposed in [1] and [43]–[48], as is shown in Table II. In order to estimate the performance of the models when forecast temperature is used, we also add a Gaussian noise with mean 0 ^{o}F, and standard deviation 1 ^{o}F to the temperature input and report the MAPE in this case. It is seen in the table that the proposed model outperforms existing models which highly depend on external feature extraction, feature selection, or hyper-parameter optimization techniques. The proposed model also has a lower increase of MAPE when modified temperature is applied. In addition, the test loss can be further reduced when more data is added to the training set.

B. Performance of the Proposed Model on the ISO-NE Dataset

The second task of the paper is to examine the generalization capability of the proposed model. To this end, we use the majority of the hyper-parameters of ResNetPlus tuned with the North-American Utility dataset to train load forecasting models for the ISO-NE dataset (The time range of the dataset is between March 2003 and December 2014). Here, the ResNetPlus structure has 10 layers on the main path.

The first test case is to predict the daily loads of the year 2006 in the ISO-NE dataset. For the proposed ResNetPlus model, the training period is from June 2003 to December 2005[6] (we reduce the size of L_h^{month} and T_h^{month} to 3 so that more training samples can be used, and the rest of the hyper-parameters are unchanged). In comparison, the similar day-based wavelet neural network (SIWNN) model in [13] is trained with data from 2003 to 2005, while the models proposed in [46] and [49] use data from March 2003 to December 2005 (both models use past loads up to 200 hours prior to the hour to be predicted). The results of MAPEs with respect to each month are listed in Table III. The MAPEs for the 12 months in 2006 are not explicitly reported in [49]. It is seen in the table that the proposed ResNetPlus model has the lowest overall MAPE for the year 2006. For some months, however, the WT-ELM-MABC model proposed in [46] produces better results. Nevertheless, as most of the hyper-parameters are not tuned on the ISO-NE dataset, we can conclude that the proposed model has good generalization capability across different datasets.

[6]The training dataset is used to determine how the snapshots are taken for the ensemble model for the ISO-NE dataset. For each implementation, 5 individual models are trained, and the snapshots are taken at 600, 650, and 700 epochs.

TABLE III
MAPES (%) OF THE PROPOSED RESNETPLUS MODEL FOR THE ISO-NE DATASET IN 2006 AND A COMPARISON WITH EXISTING MODELS

	SIWNN [13]	WT-ELM-PLSR [49]	WT-ELM-MABC [46]	Proposed model
Jan	1.60	-	**1.52**	1.619
Feb	1.43	-	**1.28**	1.308
Mar	1.47	-	1.37	**1.172**
Apr	1.26	-	**1.05**	1.340
May	1.61	-	**1.23**	1.322
Jun	1.79	-	1.54	**1.411**
Jul	2.70	-	2.07	**1.962**
Aug	2.62	-	2.06	**1.549**
Sep	1.48	-	1.41	**1.401**
Oct	1.38	-	**1.23**	1.293
Nov	1.39	-	**1.33**	1.507
Dec	1.75	-	1.65	**1.465**
Average	1.75	1.489	1.48	**1.447**

TABLE IV
COMPARISON OF THE PROPOSED RESNETPLUS MODEL WITH EXISTING MODELS ON THE ISO-NE DATASET FOR 2010 AND 2011

Model	2010	2011
RBFN-ErrCorr original [50]	1.80	2.02
RBFN-ErrCorr modified [12]	1.75	1.98
WT-ELM-PLSR [49]	**1.50**	1.80
Proposed model	**1.50**	**1.64**

We further test the generalization capability of the proposed ResNetPlus model on data of the years 2010 and 2011. The same model for the year 2006 is used for this test case, and historical data from 2004 to 2009 is used to train the model. In Table IV, we report the performance of the proposed model and compare it with models mentioned in [12], [49], and [50]. Results show that the proposed ResNetPlus model outperforms existing models with respect to the overall MAPE for the two years, and an improvement of 8.9% is achieved for the year 2011. Note that all the existing models are specifically tuned on the ISO-NE dataset for the period from 2004 to 2009, while the design of the proposed ResNetPlus model is directly implemented without any tuning.

As we use actual temperature values for the input of the proposed model (except for the "modified temperature" case of North-American Utility dataset), the results we have obtained previously provide us with an estimated upper bound of the performance of the model. Thus, we need to further analyze how the proposed model would perform when forecast temperature data is used, and whether the ensemble model is more robust to noise in forecast weather. We follow the way of modifying temperature values introduced in [43], and consider three cases of temperature modification:

- *Case 1:* add Gaussian noise with mean 0 ^{o}F, and standard deviation 1 ^{o}F to the original temperature values before normalization.

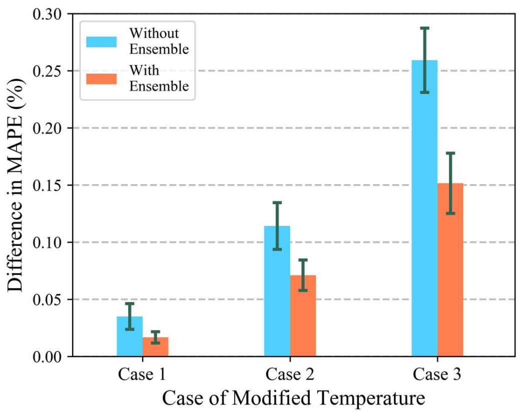


Fig. 7. The comparison of the proposed model with the ensemble strategy and the proposed model without ensemble when different cases of modified temperature are applied. The model without ensemble is a single ResNetPlus model trained with 700 epochs.

TABLE V
EMPIRICAL COVERAGES OF THE PROPOSED MODEL WITH MC DROPOUT

z-score	Expected Coverage	Empirical Coverage
1.000	68.27%	71.14%
1.280	≈ 80.00%	81.72%
1.645	≈ 90.00%	90.02%
1.960	≈ 95.00%	94.15%

- *Case 2:* add Gaussian noise with mean 0 ^{o}F, and change the standard deviation of case 1 to 2 ^{o}F.
- *Case 3:* add Gaussian noise with mean 0 ^{o}F, and change the standard deviation of case 1 to 3 ^{o}F.

For all three cases, we repeat the trials 5 times and calculate the means and standard deviations of increased MAPE compared with the case where actual temperature data is used.

The results of increased test MAPEs for the year 2006 with modified temperature values are shown in Fig. 7. We compare the performance of the proposed ResNetPlus model (which is an ensemble of 15 single snapshot models) with a single snapshot model trained with 700 epochs. As can be seen in the figure, the ensemble strategy greatly reduces the increase of MAPE, especially for case 1, where the increase of MAPE is 0.0168%. As the reported smallest increase of MAPE for case 1 in [1] is 0.04%, it is reasonable to conclude that the proposed model is robust against the uncertainty of temperature for case 1 (as we use a different dataset here, the results are not directly comparable). Is is also observed that the ensemble strategy is able to reduce the standard deviation of multiple trials. This also indicates the higher generalization capability of the proposed model with the ensemble strategy.

C. Probabilistic Forecasting for the Ensemble Model

We first use the North-American Utility dataset to demonstrate the probabilistic STLF by MC dropout. The last year of the dataset is used as the test set and the previous year is used for validation. Dropout with $p = 0.1$ is added to the previously implemented ensemble model[7] except for the input layer and the output layer (dropout with p ranging from 0.05 and 0.2 produce similar results, similar to the results reported in [38]). The first term in (8) and is estimated by a single model trained with 500 epochs (with $M = 100$ for (8) and $p = 0.1$), and the estimated value of β is 0.79.

TABLE VI
COMPARISON OF PROBABILISTIC FORECASTING PERFORMANCE MEASURES FOR THE YEAR 2011 IN THE GEFCOM2014 DATASET

Model	Pinball	Winkler (50%)	Winkler (90%)
Lasso [51]	7.44	-	-
Ind [23]	3.22	26.35	56.38
QRA [23]	2.85	25.04	55.85
Proposed model	**2.52**	**22.41**	**42.63**

The empirical coverages produced by the proposed model with respect to different *z*-scores are listed in Table V, and an illustration of the 95% prediction intervals for two weeks in 1992 is provided in Fig. 8. The results show that the proposed model with MC dropout is able to give satisfactory empirical coverages for different intervals.

In order to quantify the performance of the probabilistic STLF by MC dropout, we adopt the pinball loss and Winkler score mentioned in [23] and use them to assess the proposed method in terms of coverage rate and interval width. Specifically, the pinball loss is averaged over all quantiles and hours in the prediction range, and the Winkler scores are averaged over all the hours of the year in the test set. We implement the ResNetPlus model[8] on the GEFCom2014 dataset and compare the results with those reported in [23] and [51]. Following the setting in [23], the load and temperature data from 2006 to 2009 is used to train the proposed model, the data of the year 2010 is used for validation, and the test results are obtained using data of the year 2011. The temperature values used for the input of the model are calculated as the mean of the temperature values of all 25 weather stations in the dataset.

In Table VI, we present the values of pinball loss and Winkler scores for the proposed model and the models in [23] and [51] for the year of 2011 in the GEFCom2014 dataset. The Lasso method in [51] serves as a benchmark for methods that build regression models on the input data, and the quantile regression averaging (QRA) method in [23] builds quantile regression models on sister point forecasts (the row of Ind stands for the performance of a single model). It can be seen in Table VI that the proposed ResNetPlus model is able to provide improved probabilistic forecasting results compared with existing methods in terms of the pinball loss and two Winkler scores. As we obtain the

[7]the model implemented here uses ResNet instead of ResNetPlus, and the information of season, weekday/weekend distinction, and holiday/non-holiday distinction is not used. In addition, the activation function used for the residual blocks is ReLU.

[8]Five individual models are trained with a dropout rate of 0.1 and 6 snapshots are taken from 100 epochs to 350 epochs. M is set to 100 for MC dropout and the first term in (8) is estimated by a single model trained with 100 epochs. The estimated value of β is 0.77.

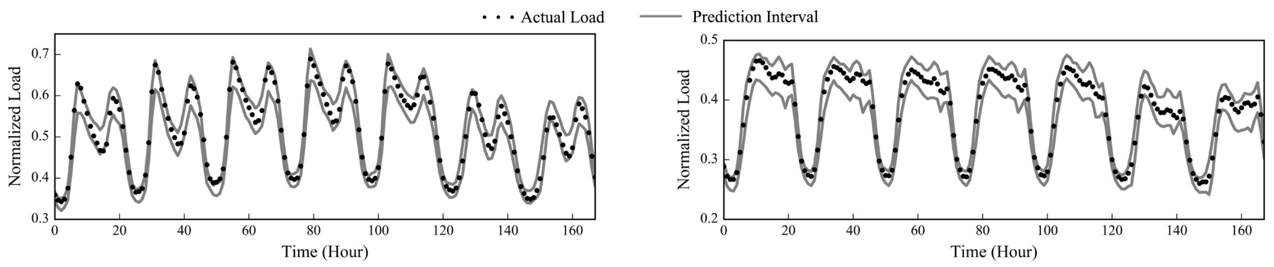


Fig. 8. Actual load and 95% prediction intervals for a winter week (left) and a summer week (right) of 1992 for the North-American Utility dataset. The two weeks start with February 3rd, 1992, and July 6th, 1992, respectively.

probabilistic forecasting results by sampling the trained neural networks with MC dropout, we can conclude that the proposed model is good at capturing the uncertainty of the task of STLF.

IV. Conclusion and Future Work

We have proposed an STLF model based on deep residual networks in this paper. The low-level neural network with the basic structure, the ResNetPlus structure, and the two-stage ensemble strategy enable the proposed model to have high accuracy as well as satisfactory generalization capability. Two widely acknowledged public datasets are used to verify the effectiveness of the proposed model with various test cases. Comparisons with existing models have shown that the proposed model is superior in both forecasting accuracy and robustness to temperature variation. We have also shown that the proposed model can be directly used for probabilistic forecasting when MC dropout is adopted.

A number of paths for further work are attractive. As we have only scratched the surface of state-of-the-art of deep neural networks, we may apply more building blocks of deep neural networks (e.g., CNN or LSTM) into the model to enhance its performance. In addition, we will further investigate the implementation of deep neural works for probabilistic STLF and make further comparisons with existing methods.

References

[1] E. Ceperic, V. Ceperic, and A. Baric, "A strategy for short-term load forecasting by support vector regression machines," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4356–4364, Nov. 2013.

[2] K.-B. Song, Y.-S. Baek, D. H. Hong, and G. Jang, "Short-term load forecasting for the holidays using fuzzy linear regression method," *IEEE Trans. Power Syst.*, vol. 20, no. 1, pp. 96–101, Feb. 2005.

[3] W. Charytoniuk, M. S. Chen, and P. V. Olinda, "Nonparametric regression based short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 13, no. 3, pp. 725–730, Aug. 1998.

[4] E. E. Elattar, J. Goulermas, and Q. H. Wu, "Electric load forecasting based on locally weighted support vector regression," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 4, pp. 438–447, Jul. 2010.

[5] J. W. Taylor, "Short-term electricity demand forecasting using double seasonal exponential smoothing," *J. Oper. Res. Soc.*, vol. 54, no. 8, pp. 799–805, Aug. 2003.

[6] M. Rejc and M. Pantos, "Short-term transmission-loss forecast for the Slovenian transmission power system based on a fuzzy-logic decision approach," *IEEE Trans. Power Syst.*, vol. 26, no. 3, pp. 1511–1521, Aug. 2011.

[7] E. A. Feinberg and D. Genethliou, "Load forecasting," in *Applied Mathematics for Restructured Electric Power Systems*, J. H. Chow, F. F. Wu, and J. Momoh, Eds. New York, NY, USA: Springer, 2005, pp. 269–285.

[8] J. W. Taylor, L. M. D. Menezes, and P. E. Mcsharry, "A comparison of univariate methods for forecasting electricity demand up to a day ahead," *Int. J. Forecast.*, vol. 22, no. 1, pp. 1–16, Jan./Mar. 2006.

[9] H. Hahn, S. Meyer-Nieberg, and S. Pickl, "Electric load forecasting methods: Tools for decision making," *Eur. J. Oper. Res.*, vol. 199, no. 3, pp. 902–907, Dec. 2009.

[10] Y. Wang, Q. Chen, T. Hong, and C. Kang, "Review of smart meter data analytics: Applications, methodologies, and challenges," *IEEE Trans. Smart Grid*, to be published, doi: 10.1109/TSG.2018.2818167.

[11] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Trans. Power Syst.*, vol. 16, no. 1, pp. 44–55, Feb. 2001.

[12] C. Cecati, J. Kolbusz, P. Różycki, P. Siano, and B. M. Wilamowski, "A novel RBF training algorithm for short-term electric load forecasting and comparative studies," *IEEE Trans. Ind. Electron.*, vol. 62, no. 10, pp. 6519–6529, Oct. 2015.

[13] Y. Chen *et al.*, "Short-term load forecasting: Similar day-based wavelet neural networks," *IEEE Trans. Power Syst.*, vol. 25, no. 1, pp. 322–330, Feb. 2010.

[14] Y. Zhao, P. B. Luh, C. Bomgardner, and G. H. Beerel, "Short-term load forecasting: Multi-level wavelet neural networks with holiday corrections," in *Proc. Power Energy Soc. Gen. Meeting*, Calgary, AB, Canada, 2009, pp. 1–7.

[15] R. Zhang, Z. Y. Dong, Y. Xu, K. Meng, and K. P. Wong, "Short-term load forecasting of Australian national electricity market by an ensemble model of extreme learning machine," *IET Gener. Transm. Distrib.*, vol. 7, no. 4, pp. 391–397, Apr. 2013.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[19] S. Ryu, J. Noh, and H. Kim, "Deep neural network based demand side short term load forecasting," *Energies*, vol. 10, no. 1, p. 3, 2016.

[20] G. Merkel, R. J. Povinelli, and R. H. Brown, "Deep neural network regression for short-term load forecasting of natural gas," in *Proc. 37th Annu. Int. Symp. Forecast.*, 2017. [Online]. Available: https://isf.forecasters.org/wp-content/uploads/ISF2017-Proceedings.pdf

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 770–778.

[22] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *Int. J. Forecast.*, vol. 32, no. 3, pp. 914–938, Jul./Sep. 2016.

[23] B. Liu, J. Nowotarski, T. Hong, and R. Weron, "Probabilistic load forecasting via quantile regression averaging on sister forecasts," *IEEE Trans. Smart Grid*, vol. 8, no. 2, pp. 730–737, Mar. 2017.

[24] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, Burlingame, CA, USA, 2016, p. 92.

[25] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Vancouver, BC, Canada, 2013, pp. 8609–8613.

[26] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.*, vol. 30, 2013, p. 3.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 1026–1034.

[28] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 972–981.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[30] K. Zhang *et al.*, "Residual networks of residual networks: Multilevel residual networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 6, pp. 1303–1314, Jun. 2018, doi: 10.1109/TCSVT.2017.2654543.

[31] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1. Honolulu, HI, USA, 2017, pp. 2261–2269.

[32] L. Zhao *et al.*, "Deep convolutional neural networks with merge-and-run mappings," *Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018.

[33] M. De Felice and X. Yao, "Short-term load forecasting with neural network ensembles: A comparative study [application notes]," *IEEE Comput. Intell. Mag.*, vol. 6, no. 3, pp. 47–56, Aug. 2011.

[34] G. Huang *et al.*, "Snapshot ensembles: Train 1, get M for free," presented at the 5th Int. Conf. Learn. Represent. (ICLR), 2017.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," presented at the 3rd Int. Conf. Learn. Represent. (ICLR), 2015.

[36] A. R. Webb, *Statistical Pattern Recognition*. Chiechester, U.K.: Wiley, 2003.

[37] L. Zhu and N. Laptev, "Deep and confident prediction for time series at Uber," in *Proc. IEEE Int. Conf. Data Min. Workshops*, New Orleans, LA, USA, 2017, pp. 103–110.

[38] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 1050–1059.

[39] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[40] F. Chollet *et al.* (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras

[41] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement.*, vol. 16. Savannah, GA, USA, 2016, pp. 265–283.

[42] T. Hong *et al.*, "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," *Int. J. Forecast.*, vol. 32, no. 3, pp. 896–913, 2016.

[43] A. J. R. Reis and A. P. A. Da Silva, "Feature extraction via multiresolution analysis for short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 20, no. 1, pp. 189–198, Feb. 2005.

[44] N. Amjady and F. Keynia, "Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm," *Energy*, vol. 34, no. 1, pp. 46–57, Jan. 2009.

[45] A. Deihimi and H. Showkati, "Application of echo state networks in short-term electric load forecasting," *Energy*, vol. 39, no. 1, pp. 327–340, Mar. 2012.

[46] S. Li, P. Wang, and L. Goel, "Short-term load forecasting by wavelet transform and evolutionary extreme learning machine," *Elect. Power Syst. Res.*, vol. 122, p. 96–103, May 2015.

[47] Z. Hu, Y. Bao, and T. Xiong, "Comprehensive learning particle swarm optimization based memetic algorithm for model selection in short-term load forecasting using support vector regression," *Appl. Soft Comput.*, vol. 25, pp. 15–25, Dec. 2014.

[48] S. Li, P. Wang, and L. Goel, "A novel wavelet-based ensemble method for short-term load forecasting with hybrid neural networks and feature selection," *IEEE Trans. Power Syst.*, vol. 31, no. 3, pp. 1788–1798, May 2016.

[49] S. Li, L. Goel, and P. Wang, "An ensemble approach for short-term load forecasting by extreme learning machine," *Appl. Energy*, vol. 170, pp. 22–29, May 2016.

[50] H. Yu, P. D. Reiner, T. Xie, T. Bartczak, and B. M. Wilamowski, "An incremental design of radial basis function networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1793–1803, Oct. 2014.

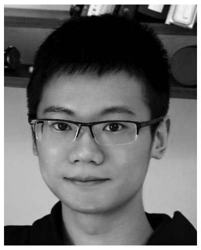
[51] F. Ziel and B. Liu, "Lasso estimation for GEFCom2014 probabilistic electric load forecasting," *Int. J. Forecast.*, vol. 32, no. 3, pp. 1029–1037, 2016.

Kunjin Chen received the B.Sc. degree in electrical engineering from Tsinghua University, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering.

His research interests include applications of machine learning and data science in power systems.

Kunlong Chen received the B.Sc. degree in electrical engineering from Beijing Jiaotong University, Beijing, China, in 2015 and the engineering degree from CentraleSupélec, Paris, France, in 2015. He is currently pursuing the M.Sc. degree with the Department of Electrical Engineering, Beijing Jiaotong University.

His research interests include applications of statistical learning techniques in the field of electrical engineering.



Qin Wang received the B.Sc. degree in electrical engineering from Tsinghua University in 2015. He is currently pursuing the master's degree with ETH Zürich, Switzerland.

His research interests include computer vision and deep learning applications.

Ziyu He received the B.Sc. degree from Zhejiang University, Hangzhou, China, in 2015 and the M.S. degree from Columbia University, NY, USA, in 2017. He is currently pursuing the Ph.D. degree with the Department of Industrial and Systems Engineering, University of Southern California, CA, USA.

His research interests are optimization and machine learning and their applications in energy.

Jun Hu (M'10) received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the Department of Electrical Engineering, Tsinghua University, Beijing, China, in 1998, 2000, and 2008, respectively.

He is currently an Associate Professor with the Department of Electrical Engineering, Tsinghua University. His research interests include overvoltage analysis in power system, sensors and big data, dielectric materials, and surge arrester technology.

Jinliang He (M'02–SM'02–F'08) received the B.Sc. degree from the Wuhan University of Hydraulic and Electrical Engineering, Wuhan, China, in 1988, the M.Sc. degree from Chongqing University, Chongqing, China, in 1991, and the Ph.D. degree from Tsinghua University, Beijing, China, in 1994, all in electrical engineering.

He became a Lecturer in 1994, and an Associate Professor in 1996, with the Department of Electrical Engineering, Tsinghua University. From 1997 to 1998, he was a Visiting Scientist with Korea Electrotechnology Research Institute, Changwon, South Korea. From 2014 to 2015, he was a Visiting Professor with the Department of Electrical Engineering, Stanford University, Palo Alto, CA, USA. In 2001, he was promoted to a Professor with Tsinghua University, where he is currently the Chair with High Voltage Research Institute. He has authored seven books and 600 technical papers. His research interests include advanced power transmission technology, sensing technology and big data mining, and smart nanodielectric materials.