Ensemble Learning for Load Forecasting

Lingxiao Wang, Shiwen Mao, *Fellow, IEEE*, Bogdan M. Wilamowski, *Life Fellow, IEEE* and R. M. Nelms, *Fellow, IEEE*

***Abstract*—In this paper, an ensemble learning approach is proposed for load forecasting in urban power systems. The proposed framework consists of two levels of learners that integrate clustering, Long Short-Term Memory (LSTM), and a Fully Connected Cascade (FCC) neural network. Historical load data is first partitioned by a clustering algorithm to train multiple LSTM models in the level-one learner, and then the FCC model in the second level is used to fuse the multiple level-one models. A modified Levenberg-Marquardt (LM) algorithm is used to train the FCC model for fast and stable convergence. The proposed framework is tested with two public datasets for short-term and mid-term forecasting at the system, zone and client levels. The evaluation using real-world datasets demonstrates the superior performance of the proposed model over several state-of-the-art schemes. For the ISO-NE Dataset for Years 2010 and 2011, an average reduction in mean absolute percentage error (MAPE) of 10.17% and 11.67% are achieved over the four baseline schemes, respectively.**

***Index Terms*—Load forecasting, deep learning, ensemble learning, long short-term memory (LSTM), smart grid, green communications.**

I. Introduction

RAPID progress in urbanization brings about significant changes in people's lifestyles. In light of this trend, many challenging problems - such as environmental pollution, traffic problems, high energy consumption, and so on - are raised. In order to address these issues, the concept of urban computing is introduced, which involves collecting, integrating, and analyzing the data generated by devices in an urban area to improve people's life quality [1], [2]. With the fast development of artificial intelligence, machine learning, in particular, deep learning, techniques show high potential for addressing many urban computing problems. This is mainly due to the breakthroughs in computing and the rapid advances in sensing and data acquisition, transmission, and storage [3]. Researchers now have the capability of handling large-scale data and utilizing it more wisely.

Today's sustainable urban power systems, i.e., the smart grid, are characterized by high energy efficiency, demand-side management, renewable energy sources, and a two-way flow of information and electricity, as enabled by the integration of communications, control, and signal processing [4]–[7]. Such work involves managing the generation and usage of electricity, as assisted by a communications network for data collection and control, to make the earth green. With the same goal of reducing energy use, the concept of green communications and networking comes out in recent years, which involves the development and application of greener and more energy-efficient communication technologies [8]. Home Area Network (HAN) and Home Energy Management (HEM) are two main applications. In HEM, at the system level, the uncertainty in power supply and demand poses one of the major challenges for energy management. Moreover, in HAN, at the client level, the deployment of renewables, such as electric vehicles (EVs) and home solar systems, brings about greatly increased randomness in the client load. A technique that can accurately predict future generation (e.g., from renewable sources) and load at both the system and client levels cooperating with energy-efficient communication technologies would be highly desirable [9]–[11], which is indispensable to achieve high power quality, save energy, and better utilize renewable energy sources and reduce costs [12].

Consequently, many methods have been proposed for load forecasting. Machine learning and statistical methods are the two main approaches that are widely applied. For example, in [13], the authors propose an ensemble approach based on extreme learning machine for short-term load forecasting. Radial Basis Function (RBF) neural networks trained with a second-order algorithm are utilized in [14] for short-term load forecasting. These two schemes both have a shallow structure in their neural network design. Deep learning has become a hot technique due to their recent demonstrated success in computer vision and natural language processing (NLP). Among various deep learning models, recurrent neural networks, e.g., Long Short Term Memory (LSTM), has been proposed for handling residential data in [15], [16]. It is shown in [15] that an LSTM-based Sequence to Sequence (S2S) architecture can handle both one-minute and one-hour resolution data for one residential customer. In [16], the authors focus on short-term forecasting individual customer's consumption of power using LSTM. Effectiveness of accurate short-term load forecasting has been demonstrated in [17] by using a Deep Residual Network (res-net). In addition, Quantile Regression

Manuscript received September 20, 2019; revised February 20, 2020; accepted March 30, 2020. Date of publication April 13, 2020; date of current version May 19, 2020. This work was supported in part by the U.S. National Science Foundation under Grant DMS-1736470. The associate editor coordinating the review of this article and approving it for publication was E. Ayanoglu. This article was presented in part at IEEE GreenCom 2019, Atlanta, GA, USA, July 2019. *(Corresponding author: Shiwen Mao.)*

Lingxiao Wang, Shiwen Mao, and R. M. Nelms are with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: lzw0039@auburn.edu; smao@ieee.org; nelmsrm@auburn.edu).

Bogdan M. Wilamowski is with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA, and also with the Department of Electronics and Telecommunications, University of Information Technology and Management, 35-225 Rzeszów, Poland (e-mail: wilambm@auburn.edu).

Digital Object Identifier 10.1109/TGCN.2020.2987304

is a popular statistic technique for load forecasting. In [18], the authors exploit the quantile regression model to enhance forecasting performance. In [19], the authors improve the traditional quantile regression neural network and demonstrate its reliability in probabilistic load forecasting.

In this paper, an ensemble learning approach is proposed to tackle the load forecasting problem. Our proposed framework consists of two levels of learners. The first-level learner utilizes the LSTM model to obtain the first-level predictions, while a fully connected cascade (FCC) neural networks are incorporated in the second-level learner for the purpose of model fusion. Our proposed framework has three notable features. First, point load forecasting is a regression problem, to which unsupervised learning techniques can be easily applied. The proposed framework integrates unsupervised learning with a supervised learning model for accurate load prediction, which is a novel approach comparing to existing load forecasting models. Specifically, clustering algorithms are incorporated in our framework, to partition data into individual clusters according to their similarity. Each data cluster is then used to generate an LSTM base model to obtain the first-level prediction. Then the first-level prediction results are fused by the second-level FCC neural network as supervised learning to enhance the accuracy of load forecasting.

Second, for various learning problems, a deep neural network may not always be the chosen one; it is critical to choose the right neural network structure properly. In this work, we select a deep (LSTM) and a shallow (FCC) structure in the two different levels of learning, respectively. It is well-known that the deeper the neural network, the more likely overfitting will occur. Thus, it is highly desirable to have a learner that can provide a sufficient learning ability, while using as few layers as possible. In the proposed framework, the first-level learner captures most of the nonlinear relationship between input and output data, while the second-level learner discerns the linear connection between them. This is the criterion that guides our choice of proper neural architecture in the proposed framework. Third, ensemble learning is used in the proposed framework. The boosted fusion model (ensemble) in the second level enhances the accuracy of load prediction [20].

Our contributions in this work can be briefly summarized as follows. First, an ensemble learning approach is proposed to integrate state-of-the-art machine learning algorithms, i.e., clustering, LSTM, and FCC, for accurate load forecasting. We also study four different, representative clustering algorithms applied in the first level of learning and found the integration of HDBSCAN and LSTM achieve the best performance. Second, we propose to use an FCC neural network for model fusion in the second-level learner and a fast converging and stable modified Levenberg-Marquardt (LM) optimization algorithm for training the second-level learner. The FCC network captures the relationship among individual models and thus improve the prediction accuracy. Third, we validate our proposed framework with two public datasets and compare its performance with several state-of-the-art schemes, where superior performance is demonstrated for the proposed framework. Fourth, the proposed framework can effectively deal with both short-term (e.g., hour-ahead) and mid-term (e.g., week-ahead) load forecasting, for not only system-level but also zone-level and client/residential-level forecasting.

The remainder of this paper is organized as follows. In Section II, we describe our proposed framework. We then discuss optimization and training in Section III. Experimental validation of the proposed framework is presented in Section IV. Section V concludes this paper.

II. The Proposed Framework

In this section, we first formulate the power load forecasting problem. We then discuss the details of our proposed framework in the remainder of the section, including the design of the two levels of learners.

A. Problem Statement

In this paper, we focus on the load forecasting problem. Consider a time series signal $\boldsymbol{Y}_T = \{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_{m-1}, \boldsymbol{\ell}\}$, where $\boldsymbol{Y}_T \in \mathbb{R}^{m \times T}$. $\boldsymbol{Y}_T$ consists of two components, i.e., the feature part and load part. In the feature part, $\boldsymbol{f}_i = \{f_{i1}, f_{i2}, \ldots, f_{iT}\}$, which is the historical data of the ith feature that affects load. For example, temperature is one of the most important features that affect the power load. If features are not provided in the dataset, this part would set to null, and the forecasting will use historical load data only. The load part consists of $\boldsymbol{\ell} = \{\ell_1, \ell_2, \ldots, \ell_T\}$, i.e., the historical load data.

The goal is to forecast the load at a future time $T + \tau$ in a rolling predicting fashion, where τ is the amount of time ahead of the current time T. That is, we assume that only the information at and before T, i.e., $\boldsymbol{Y}_t$, for $t \leq T$, is available when predicting $\ell_{T+\tau}$. For example, to forecast the load value at time $T + 1$ (i.e., one time step ahead), $\boldsymbol{Y}_T$ is available and used. In order to ease training and reduce the training time, a window filter W is applied to $\boldsymbol{Y}_T$, which stores only the data for w time steps, from the current time T back to time $T - w + 1$. The input matrix $\boldsymbol{S}_T$ is thus defined as $\boldsymbol{S}_T = W(\boldsymbol{Y}_T)$, which is an $m \times w$ matrix.

Fig. 1 presents the mechanism of window filter and the formation of input and output data. The forecast value $\hat{\ell}_{T+\tau}$ is obtained by a fitting function as

$$\hat{\ell}_{T+\tau} = g(\boldsymbol{S}_T). \tag{1}$$

The goal of our proposed machine learning based predictive method is to learn the fitting function $g(\cdot)$ from the dataset $\boldsymbol{Y}_T$ that is available.

B. The Proposed Ensemble Learning Framework

To achieve high accuracy of power load forecasting, the concept of *stacking* is incorporated in our framework [21]. Stacking is a procedure of first training individual machine learning models and then integrating them [20]. There are two levels of learners in our proposed framework, where the first-level learner consists of multiple *individual learning models* and the second-level learner is used to combine the outputs from the individual learners in the first level for an integrated output. In order to meet the feature of stacking and testing, the data should first be divided into three parts. The first-level

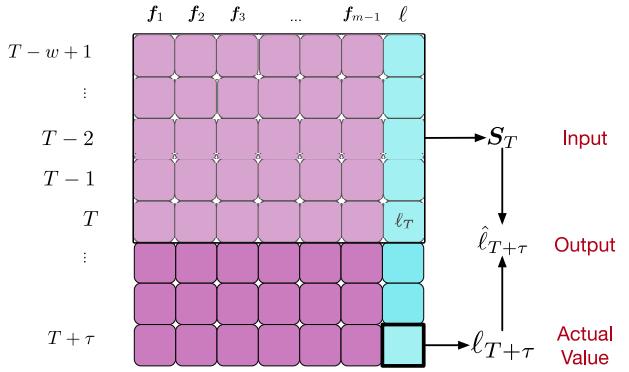


Fig. 1. Input and output of the proposed neural network model.

Algorithm 1: Small Build the First Level LSTM Predictors

1 Partition the input data in dataset $D1$ into k clusters using a clustering algorithm;
2 Divide dataset $D1$ into k individual datasets (i.e., including both input/output data) according to the clustering results: $\{\mathrm{D1}_1, \mathrm{D1}_2, ..., \mathrm{D1}_k\}$, where $\mathrm{D1}_i$ is the ith dataset produced by the ith cluster;
3 Use each dataset $\mathrm{D1}_i$, $i = 1, 2, ..., k$ to train an individual LSTM model i;

learners use the first part of data (denoted by $D1$). After the First-level learning models are built and trained, new data are generated from this level of learner, which is combined with the second and third parts of data (denoted by $D2$ and $D3$, respectively). The combined two parts of data are used to train the second-level learner and test the framework.

In this paper, we propose to use LSTM a recurrent neural network model, for the first-level learning and the FCC neural network for second-level learning. Fig. 2 illustrates the structure of the proposed framework. After preprocessing, the dataset is clustered into three parts, $D1$, $D2$, and $D3$ for training and testing purposes. The proposed predictor consists of a clustering algorithm, a set of LSTM models in the first-level learner, and an FCC model in the second-level learner. We discuss the design of these components in detail in the rest of this section.

C. First Level Learner

The first-level learner consists of a set of LSTM predictive models as well as a clustering algorithm, whose procedure is presented in Algorithm 1. The clustering algorithm partition the input data $D1$ into $D1_1, D1_2, \ldots, D1_k$, each being used to train an individual LSTM model.

1) Clustering: Before data can be used by the LSTM models, we employ a clustering algorithm to partition the dataset based on the similarity among input data samples. Clustering is usually an unsupervised machine learning technique, referring to the process of grouping unlabeled data into clusters of similar features [22].

Note this is different from classification, which is based on given labeled data. It is well-known that the electricity demand is correlated with various obvious factors, such as temperature and calendar dates (e.g., weekday, holiday, month, season, etc.), while also being affected by uncertainties or latent factors as well.

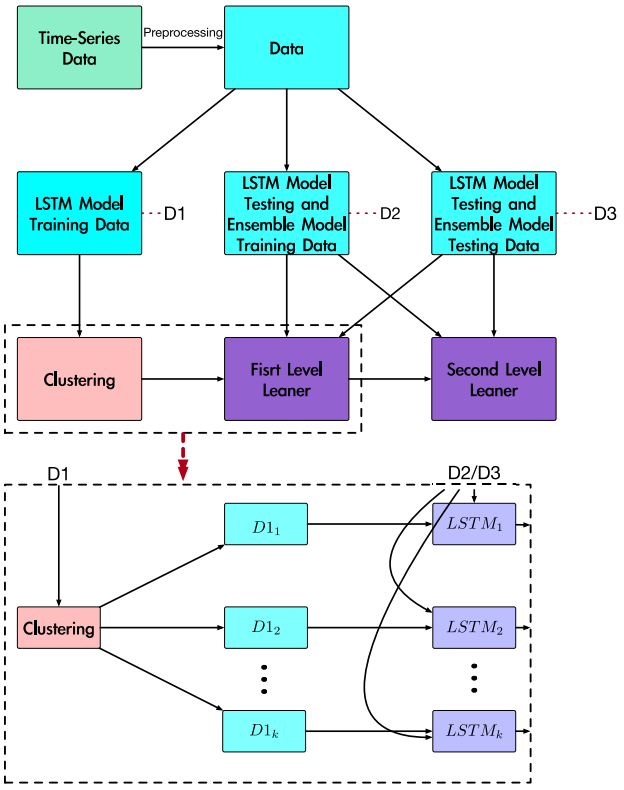


Fig. 2. The proposed load prediction framework with two levels of learners.

We propose the use of unsupervised learning in our forecasting model with the following reasons. First of all, group input data of load forecasting into suitable sets and use different learning model for each set, are beneficial to better explore the correlation in the dataset [11]. Second, we assume that short term load variations are affected by the historical data of the time immediately before the current time. With unlabeled historical electric load data, clustering can group the data samples automatically and reasonably. Last but not least, partitioning the training dataset first and combining the learning results from the models later, resembles a kind of resampling process. This is similar to the process of cross-validation technique, which can mitigate the overfitting problem in machine learning.

2) Clustering Algorithms: The collected power load time series data is usually susceptible to noise, shifting, and deformation [23]. It is important to choose an appropriate clustering method, from various existing techniques, to handle such data. In this paper, we choose four representative algorithms from three categories of clustering methods, i.e., (i) partitioning methods, (ii) hierarchical methods, and (iii) density based methods. The chosen methods are K-means++ [24], BIRCH [25], DBSCAN [26], and HDBSCAN [27], [28], as summarized in Table I. Note that for DBSCAN and HDBSCAN, some data samples are identified as outliers. Such group of outlier data is treated as one unique cluster in our proposed framework.

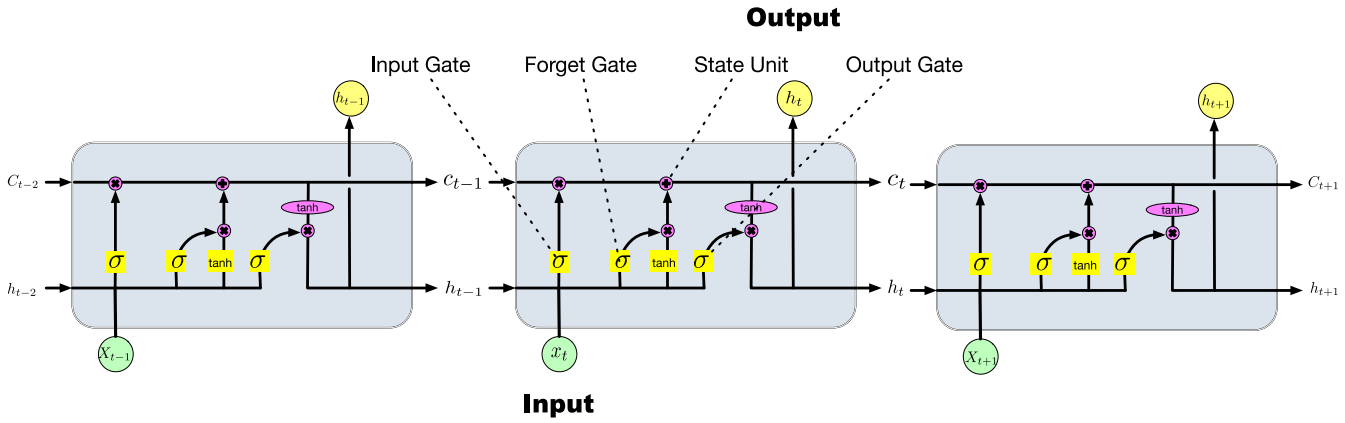


Fig. 3. An unfolded view of the LSTM neural cell structure.

TABLE I
CLUSTERING ALGORITHMS USED IN THIS PAPER

Partitioning	
K-Means++ [24]	1.1 Choose seeds (i.e., the initial cluster centers) for K-means 1.2 Improve speed and accuracy of K-means
Hierarchical	
BIRCH [25]	2.1 Balanced Iterative Reducing and Clustering using Hierarchies 2.2 Based on the concept of Clustering Feature (CF) and CF tree 2.3 Does not need a predetermined number of clusters k 2.4 Can remove noise (outliers)
Density Based	
DBSCAN [26]	3.1 Density Based Spatial Clustering of Application with Noise 3.2 Uses parameters $(\epsilon, Minpts)$ to characterize the density of the data space
HDBSCAN [27], [28]	4.1 Hierarchical DBSCAN 4.2 Removes border points in DBSCAN 4.3 Superior to DBSCAN from a qualitative clustering perspective [29]

3) Long Short-Term Memory (LSTM): Inspired by the novel idea of using three types of gates to regulate information flow and remembering information for over an arbitrary time interval [30], LSTM overcomes the limitation of long memory capability in recurrent neural networks. An unfolded illustration of the LSTM neural network is presented in Fig. 3. Input gate i_t, forget gate f_t, output gate o_t, and state unit c_t are the four key components in each LSTM cell (for time t). The state of LSTM cell at time t is calculated as

$$i_t = \sigma\Big(\boldsymbol{W}^i h_{t-1} + \boldsymbol{U}^i x_t + \boldsymbol{b}^i\Big) \tag{2}$$

$$f_t = \sigma\Big(\boldsymbol{W}^f h_{t-1} + \boldsymbol{U}^f x_t + \boldsymbol{b}^f\Big) \tag{3}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \sigma(\boldsymbol{W} h_{t-1} + \boldsymbol{U} x_t + \boldsymbol{b}) \tag{4}$$

$$o_t = \sigma(\boldsymbol{W}^o h_{t-1} + \boldsymbol{U}^o x_t + \boldsymbol{b}^o) \tag{5}$$

$$h_t = \tanh(c_{t-1}) \cdot o_t. \tag{6}$$

In the training phase, each LSTM model $LSTM_i$ will be trained with the corresponding data cluster $D1_i$, $i = 1, 2, \ldots, k$, as shown in Fig. 2.

4) Testing Process in the First Level Learner: During the training phase for the level two learner and the testing phase, new input data samples beyond $D1$ (i.e., in $D2$ and $D3$, respectively) arrives and are fed into the first-level learner. How to deal with them should be carefully designed. One way is to select the most similar cluster and use the corresponding trained LSTM model as in our prior work [11]. In this paper, however, we propose to use *ensemble learning*, which is based on the assumption that power load prediction is driven by each of the homogeneous first level models. Thus the new data sample is fed into each first-level LSTM model, and an FCC neural network is used in the second level to fuse the outputs from the LSTM models to produce a single prediction.

D. Second Level Learner

Dataset $D2$ is used to train the second-level learner. Specifically, the data samples in $D2$ are first fed into each trained LSTM predictors in the first-level. Each LSTM predictor then generates a prediction value. These outputs are used as input to train the second-level learner.

The FCC neural network is incorporated for ensemble learning at level two. Fig. 4 shows an example of the FCC ensemble neural network. In this example, k base models are available and to be fused by five neurons. The first four neurons are activated by the tanh$(\cdot)$ activation function, given by $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$. The last neuron is a linear summation. With the same number of neurons in level two, the FCC neural network architecture is superior to traditional neural network structures [31], as it provides more connections (and weights) than the traditional architecture, which make it deeper. The FCC neural network is similar to Deep Residual Networks [32] in some sense, which has an identity mapping for every input and latent variable to every neuron.

III. OPTIMIZATION AND TRAINING

A. Problem Formulation

We use the sum square error as the default lost function for the two levels of learners. The corresponding objective function of the LSTM model i at lever one is defined as

$$\mathcal{L}(L_1; i) = \underset{\boldsymbol{\omega}^i_{lstm}}{\text{minimize}} \sum_{T \in D1_i} \left\| \hat{\ell}^{L_1,i}_{T+\tau} - \ell_{T+\tau} \right\|^2 + \alpha \cdot \left\| \boldsymbol{\omega}^i_{lstm} \right\|, \tag{7}$$

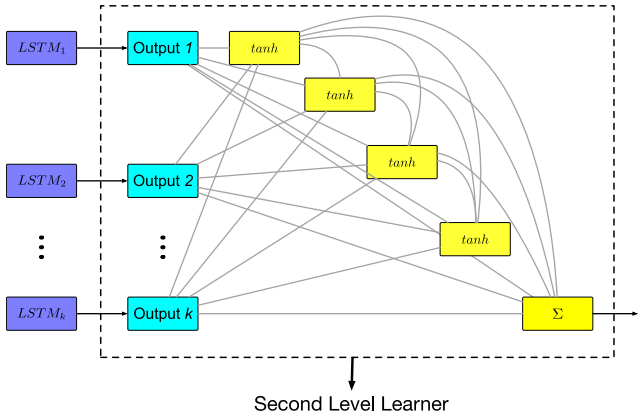


Fig. 4. An example of FCC ensemble neural network used in the second-level learner.

where $\hat{\ell}^{L_1,i}_{T+\tau}$ is the predicted value of load by LSTM model i for time $T+\tau$, $\ell_{T+\tau}$ is the ground truth (i.e., label), and $\boldsymbol{\omega}^i_{lstm}$ are the wights of LSTM model i at the first level.

Supposing there are k trained LSTM models in the first-level learner, the load predicted by the level-two learner at time $T+\tau$ is given by

$$\hat{\ell}^{L_2}_{T+\tau} = f\Big(\hat{\ell}^{L_1;1}_{T+\tau}, \hat{\ell}^{L_1;2}_{T+\tau}, \ldots, \hat{\ell}^{L_1;k}_{T+\tau}; \boldsymbol{\omega}_{fcc}\Big), \tag{8}$$

where $f()$ is the output of the ensemble FCC neural network, $\hat{\ell}^{L_1;i}_{T+\tau}$ is the load forecast value predicted by LSTM model i, and $\boldsymbol{\omega}_{fcc}$ are the weights of the ensemble FCC neural network. The corresponding optimization objective over the validation and ensemble dataset D2 in level two is given by

$$\mathcal{L}(L_2) = \underset{\boldsymbol{\omega}_{fcc}}{\text{minimize}} \sum_{T\in D2} \left\| \hat{\ell}^{L_2}_{T+\tau} - \ell_{T+\tau} \right\|^2 + \beta \cdot \|\boldsymbol{\omega}_{fcc}\|. \tag{9}$$

In both the first-level and second-level optimization objective functions, the L1 regulation is used to prevent overfitting in the neural network training process.

B. Gradient Descent Algorithms

First-order gradient descent algorithms, such as error back propagation, Stochastic Gradient Decent (SGD), and its variants Adam, are quite successful in training deep neural networks. However, ill-conditioning and local-minima are common challenges for these algorithms. In [33], a second-order gradient descent algorithm is proved as an effective solution for optimizing problems with an objective function that exhibits pathological curvature. However, the second-order gradient descent algorithm also has its limitations. One challenge is that, for very deep neural networks, the second-order algorithm calculates the Hessian Matrix of the neural network, which takes a relatively longer period of time to train. The other issue is that, as the number of layers is increased, the large values of weights may get stuck in the saturated region, whose derivative of gradient tends to zero, and thus causing a vanishing gradient condition (known as the flat-spot problem) [34].

Algorithm 2: The Modified Levenberg MarQuardt Method

1 Set $0 < m < \alpha_1$ and $0 < p_0 < p_1 < p_2 < 1$, where $\alpha_1 = 10^{-6}$, $m = 10^{-7}$, $p_0 = 10^{-4}$, $p_1 = 0.2$, $p_2 = 0.8$, and $e = 1$;
2 Calculate Jacobian Matrix $\boldsymbol{J}(\boldsymbol{\omega}^e_{fcc})$ and approximate the Hessian matrix of the FCC neural network at the second level at iteration $e = 1$;
3 The normal LM step as $\boldsymbol{d}_e = \Delta\boldsymbol{\omega}^e_{fcc}$;
4 A line search for approximating the LM step $\Delta\boldsymbol{\omega}^{e'}_{fcc}$;
5 Combine Steps 2 and 3 as $\boldsymbol{s}_e = \Delta\boldsymbol{\omega}^e_{fcc} + \alpha_e \Delta\boldsymbol{\omega}^{e'}_{fcc}$;
6 If $\boldsymbol{J}(\boldsymbol{\omega}^e_{fcc})^T \boldsymbol{J}(\boldsymbol{\omega}^e_{fcc}) = 0$, then stop;
7 Compute $r_e = R^e_a / R^e_p$, and set

$$\boldsymbol{\omega}^{e+1}_{fcc} = \begin{cases} \boldsymbol{\omega}^e_{fcc} + \boldsymbol{s}_e, & \text{if } r_e > p_0 \\ \boldsymbol{\omega}^e_{fcc}, & \text{otherwise;} \end{cases} \tag{10}$$

8 Compute

$$\alpha_{e+1} = \begin{cases} 4\alpha_e, & \text{if } r_e < p_1 \\ \alpha_e, & \text{if } r_e \in [p_1, p_2] \\ \max(0.25\alpha_e, m), & \text{if } r_e > p_1; \end{cases} \tag{11}$$

9 Set $e = e + 1$, and go to Step 2;

Given all the advantages and disadvantages of second-order gradient descent algorithms, we choose to apply the Adam algorithm [35], which is a first order gradient-based algorithm, to solve the regression task problem at level one, due to its deep structure. At level two, where FCC is a shallow neural network, we utilize the modified Levenberg-Marquardt (LM) Algorithm [36], which is a second-order optimization algorithm. The reason for a shallow architecture is applied at level two is that, we aim to provide a sufficient learning capacity for the training samples with the least number of neurons to overcome the overfitting problem.

C. Modified Levenberg-Marquardt (LM) Algorithm

In this section, we introduce how to apply the modified LM in training the ensemble neural network at level two. The procedure is presented in Algorithm 2. The convergence of this method is proven in [36], [37].

The Jacobian Matrix $\boldsymbol{J}(\boldsymbol{\omega}^e_{fcc})$ at iteration e is calculated by the derivative of (9), which is given by

$$\boldsymbol{J}\Big(\boldsymbol{\omega}^e_{fcc}\Big) = \Big[\frac{\partial\mathcal{L}(L_2)}{\partial\omega^e_1}, \frac{\partial\mathcal{L}(L_2)}{\partial\omega^e_2}, \ldots, \frac{\partial\mathcal{L}(L_2)}{\partial\omega^e_Z}\Big], \tag{12}$$

where $\boldsymbol{\omega}^e_{fcc}$ is the weights of the FCC neural network at iteration e, which has Z weight values denoted by $\{\omega^e_1, \omega^e_2, \ldots, \omega^e_Z\}$. The Hessian matrix can be approximated by $\boldsymbol{J}(\boldsymbol{\omega}^e_{fcc})\boldsymbol{J}(\boldsymbol{\omega}^e_{fcc})^T$. A damping factor μ_e is updated iteratively as

$$\mu_e = \alpha_e \Big\| \mathcal{L}\Big(L_2, \boldsymbol{\omega}^e_{fcc}\Big) \Big\|^\beta, \tag{13}$$

where $\beta \in (0, 2]$. At each iteration, the weights of the FCC neural network are updated as

$$\boldsymbol{\omega}^{e+1}_{fcc} = \boldsymbol{\omega}^e_{fcc} + \boldsymbol{s}_e, \tag{14}$$

or

$$\boldsymbol{\omega}_{fcc}^{e+1} = \boldsymbol{\omega}_{fcc}^{e} + \Delta\boldsymbol{\omega}_{fcc}^{e} + \alpha_e \Delta\boldsymbol{\omega}_{fcc}^{e'}, \tag{15}$$

where $\boldsymbol{s}_e \doteq \Delta\boldsymbol{\omega}_{fcc}^{e} + \alpha_e \Delta\boldsymbol{\omega}_{fcc}^{e'}$; $\boldsymbol{d}_e \doteq \Delta\boldsymbol{\omega}_{fcc}^{e} = -[\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e})^T \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e}) + \mu_e \boldsymbol{I}]^{-1}\ \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e})^T\ \mathcal{L}(L_2, \boldsymbol{\omega}_{fcc}^{e})$ is the normal LM step; $\Delta\boldsymbol{\omega}_{fcc}^{e'}$ is a line search for approximating the LM step, which is defined as

$$\Delta\boldsymbol{\omega}_{fcc}^{e'} = -\Big[\boldsymbol{J}\big(\boldsymbol{\omega}_{fcc}^{e} + \Delta\boldsymbol{\omega}_{fcc}^{e}\big)^T \boldsymbol{J}\big(\boldsymbol{\omega}_{fcc}^{e} + \Delta\boldsymbol{\omega}_{fcc}^{e}\big) + \mu_e' \boldsymbol{I}\Big]^{-1} \times \boldsymbol{J}\big(\boldsymbol{\omega}_{fcc}^{e} + \Delta\boldsymbol{\omega}_{fcc}^{e}\big)^T \mathcal{L}\big(L_2, \boldsymbol{\omega}_{fcc}^{e} + \Delta\boldsymbol{\omega}_{e}\big), \tag{16}$$

where $\mu_e' = \|\mathcal{L}(L_2, \boldsymbol{\omega}_{fcc}^{e} + \Delta\boldsymbol{\omega}_{fcc}^{e})\|^{\beta}$, α_e is a parameter iterative updated as in (11) in Algorithm 2; $\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e} + \Delta\boldsymbol{\omega}_{fcc}^{e})$ is approximated by $\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e})$; and μ_e' is approximated by μ_e for reducing the computational overhead. Then we can rewrite (16) as

$$\Delta\boldsymbol{\omega}_{fcc}^{e'} = -\Big[\boldsymbol{J}\Big(\boldsymbol{\omega}_{fcc}^{e}\Big)^T \boldsymbol{J}\Big(\boldsymbol{\omega}_{fcc}^{e}\Big) + \mu_e \boldsymbol{I}\Big]^{-1} \times \boldsymbol{J}\Big(\boldsymbol{\omega}_{fcc}^{e}\Big)^T \mathcal{L}\Big(L_2, \boldsymbol{\omega}_{fcc}^{e} + \Delta\boldsymbol{\omega}_{fcc}^{e}\Big). \tag{17}$$

In order to justify whether $\boldsymbol{s}_e$ is a good step or not, the trust region technique is used. The actual reduction R_a^e and the newly predicted reduction R_p^e at the eth iteration are defined in (18) and (19), respectively.

$$R_a^e = \Big\|\mathcal{L}\Big(L_2, \boldsymbol{\omega}_{fcc}^{e}\Big)\Big\|^2 - \Big\|\mathcal{L}\Big(L_2, \boldsymbol{\omega}_{fcc}^{e} + \boldsymbol{s}_e\Big)\Big\|^2 \tag{18}$$

$$R_p^e = \Big\|\mathcal{L}\Big(\boldsymbol{\omega}_{fcc}^{e}\Big)\Big\|^2 - \Big\|\mathcal{L}\Big(\boldsymbol{\omega}_{fcc}^{e}\Big) + \boldsymbol{J}\Big(\boldsymbol{\omega}_{fcc}^{e}\Big)\boldsymbol{d}_e\Big\|^2 + \Big\|\mathcal{L}\Big(\boldsymbol{\omega}_{fcc}^{e} + \boldsymbol{d}_e\Big)\Big\|^2 - \Big\|\mathcal{L}\Big(\boldsymbol{\omega}_{fcc}^{e} + \boldsymbol{d}_e\Big) + \alpha_e \boldsymbol{J}\Big(\boldsymbol{\omega}_{fcc}^{e}\Big)\Delta\boldsymbol{\omega}_{fcc}^{e'}\Big\|^2. \tag{19}$$

Their values are then compared by $r_e = R_a^e / R_p^e$, and the weights are updated according to the value of r_e as in (10) in Algorithm 2.

IV. Evaluation With Real-World Datasets

Extensive experiments of load forecasting are conducted on two datasets at the system level and the residential level, respectively, to validate the performance of the proposed ensemble learning framework. The proposed framework is implemented with Keras 2.2.4, TensorFlow 2.0-beta, and Sklearn 0.20.0 in the Python 3.7 environment. The neural network for model fusion at level two is implemented using ADNBN coded by us using MATLAB R2018a.

A. Datasets

1) Dataset Description: The following two public benchmark datasets are used for performance evaluation.

- *The ISO-NE dataset [38]:* This is a collection of hourly temperature and load data over 12 years from Jan. 1, 2007 to Dec. 31, 2018 in the New England area, including data for each of the eight zones (i.e., Connecticut-CT, Maine-ME, New Hampshire-NH, Rhode Island-RI, Vermont-VT, Massachusetts of NEM-NEMASS, Massachusetts of SEM-SEMASS, and Massachusetts of WC-WCMASS) and for the entire ISO-NE transmission system. Fig. 5 presents the entire system level load and temperature data of the ISO-New England dataset in 2018. The load of each of the eight zones in 2018 is plotted in Fig. 6.
- *The Residential Electricity Consumption dataset [39]:* This is a collection of 370 clients' electricity consumption recorded for every 15 minutes during a period of three years from 2011 to 2014. Portuguese clients can be either residential or industrial consumers. Note that we only use the data for 320 clients, as the data for the remaining 50 clients are collected after 2011 (i.e., incomplete).

TABLE II
The Search Spaces of Algorithm Parameters

Algorithms	Parameters	Search Space
K-means++	number of clusters	[2:1:20]
Birch	number of clusters	[2:1:20]
DBSCAN	maximum distance between samples	[.5:.05:.8]
	minimum number of samples	[5:5:30]
HDBSCAN	minimum number of samples	[5:5:30]
LSTM	number of hidden neurons	[16,32,64,128]
	learning rate	[0.001,0.05,0.01]
	training epochs	[50:50:200]
FCC	number of hidden neurons	[2:1:11]
	training epochs	[50:50:150]
	activation function	[tanh,sigmoid,ReLU]

2) Preprocessing: A sliding window technique of P samples is implemented on historical time-series dataset during the training process. The period of P is divided into three parts, as shown in Fig. 2. The ratio of split is 2:1:1. For example, if hourly day-ahead load of Year the 2017 is predicted, the period P is set to 4 years. The data for one year from 2014 to 2015 partitioned to dataset $D1$, the data for 2016 and 2017 become $D2$ and $D3$, respectively. When forecasting the load for the Year 2018, P is chosen from 2015 to 2018.

Normalization is applied in the preprocessing process. As shown in [40]–[42], normalization can not only speed up the convergence of training, but also reveal the true similarity between time series data. In order to prevent data snooping in time series prediction, which makes use of future information to enhance performance of forecast, only datasets $D1$ and $D2$ are normalized. In the testing set $D3$, new data generated by the first-level learner is restored from normalized form to the original form. The definition of normalization is

$$\boldsymbol{S}_{T;i}^{norm} = \frac{\boldsymbol{S}_{T;i} - \min(\boldsymbol{S}_{T;i})}{\max(\boldsymbol{S}_{T;i}) - \min(\boldsymbol{S}_{T;i})}, \tag{20}$$

where $\boldsymbol{S}_{T;i}^{norm}$ and $\boldsymbol{S}_{T;i}$ are the normalized and original form of data sample i in dataset $\boldsymbol{S}_T$, respectively.

B. Experiments and Results

In our experiments, the grid search technique is applied for hyper-parameters tuning. The search space for the parameters in each machine learning algorithm is presented in Table II.

1) System Level Prediction Performance: At the overall system level, short and mid term load forecasting are conducted on the ISO-NE dataset. The *first case* we examine is

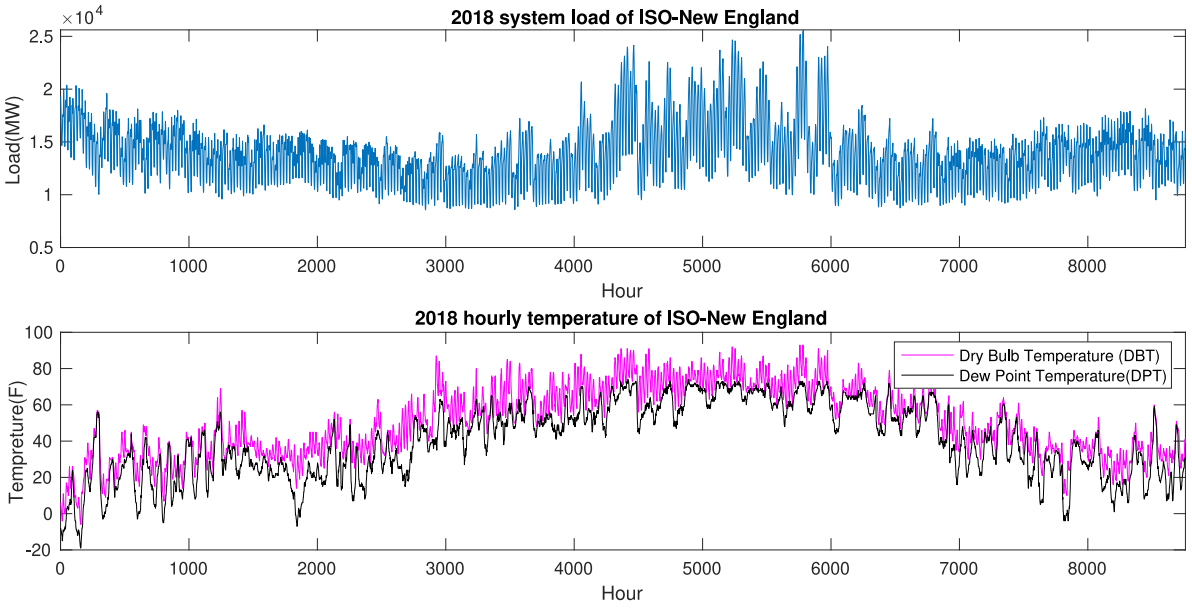


Fig. 5. The 2018 overall system level load and temperature of the ISO-New England dataset.

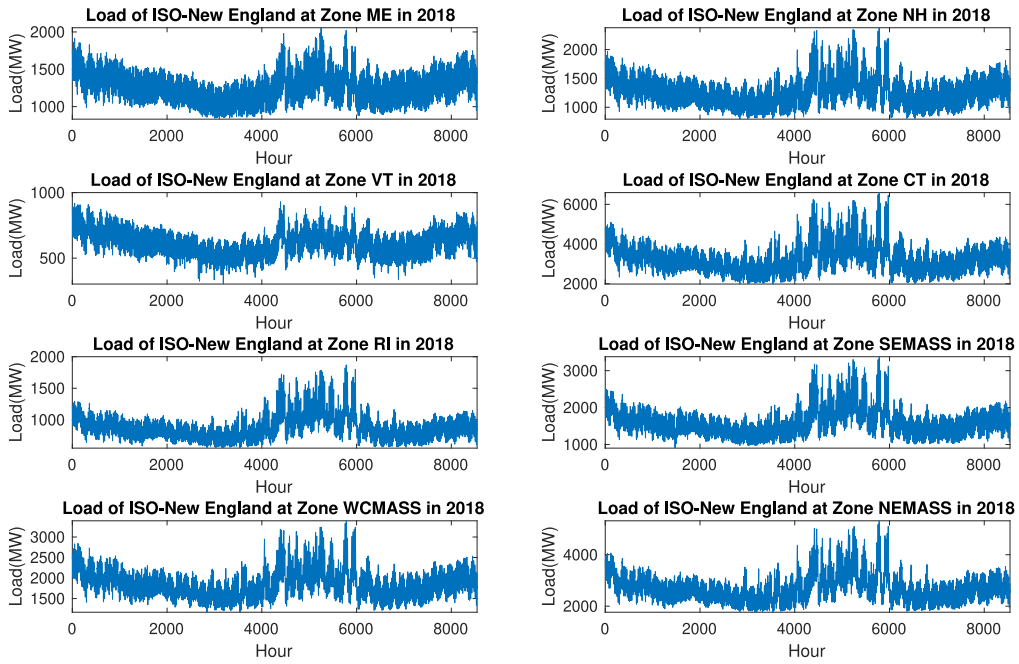


Fig. 6. The individual load for each of the eight Zones in Year 2018 of the ISO-New England dataset.

short-term forecasting, which predicts the load of the next day 24-hours ahead. In order to compare our method's performance with the existing cutting-edge technique, the system load in the Year 2010 and 2011 of ISO-NE are predicted individually, each using the three previous years' data as training and ensemble learning (see Section IV-A2). We utilize the similar inputs as in [17]. Table III summarized the input of this case. For $feature_9$, the actual value of the temperature of the next day is used in all the schemes, based on the assumption that this information is available and the fact weather forecast is extremely accurate now-days.

Three state-of-the-art models proposed in [13], [14], [17] and the traditional LSTM recurrent neural network model are used as benchmarks for comparison with our proposed framework. The performance results in the form of mean absolute percentage error (MAPE) are shown in Table IV. The number of first-level learners in our proposed module is presented in the second column for each year as well. The table shows that the four variants of our proposed framework all outperform the four benchmark schemes. An average reduction in MAPE of 10.17% in the Year 2010 and 11.67% in the Year 2011 are achieved over the four baseline schemes.

We also find that the HDSCAN based approach outperforms the other variants of our framework. To illustrate the efficacy of ensemble learning, we also present the performance of the first-level and second-level learners in Table V. The table

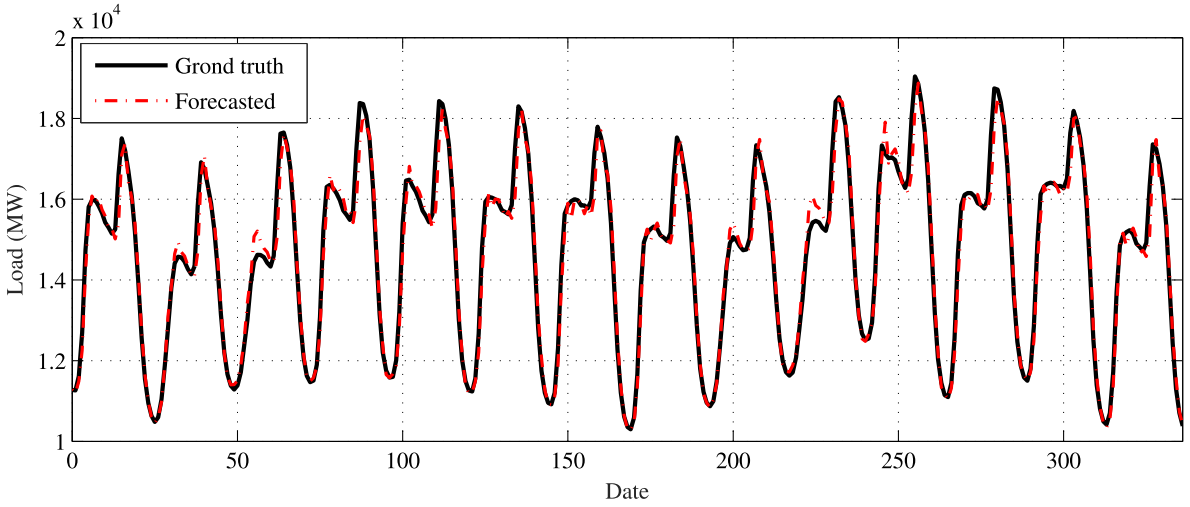


Fig. 7. System load forecast results for the last two weeks of 2011 on the ISO-NE dataset using the HDBSCAN-LSTM model.

TABLE III
INPUT DATA AND OUTPUT FOR SHORT-TERM LOAD FORECASTING AT TIME h

Input	
$feature_1$	Load of the hth hour of the day that are 1, 2, 3, 4 and months prior to the next day
$feature_2$	Load of the hth hour of the day that are 1, 2, 3, 4 weeks prior to the next day
$feature_3$	Load of the hth hour of the day that are 1 days prior to the next day
$feature_4$	Load of the most recent 24 hours prior to the hth hour of the next day
$feature_5$	Temperature of the same hour as $feature_1$
$feature_6$	Temperature of the same hour as $feature_2$
$feature_7$	Temperature of same hour as $feature_3$
$feature_8$	Temperature of the hth hour of the next day
$feature_9$	Indicator (1,0) for season (winter, summer), weekend, and holiday
ℓ_h	Load at time h
Output	
$\hat{\ell}_{h+24}$	24 hours ahead load, i.e., $\tau = 24$

TABLE IV
COMPARISON OF PROPOSED MODEL WITH OTHER MODELS USING THE ISO-NE DATASET FOR YEARS 2010 AND 2011

	ISONE (SYS) 2010		ISONE (SYS) 2011	
Model's	MAPE	Number	MAPE	Number
ErrCorr modified [14]	1.75	-	1.98	-
ELM-PLSR [13]	**1.50**	-	1.80	-
DRN [17]	**1.50**	-	1.64	-
LSTM	1.58	-	**1.50**	-
K-means++-LSTM	1.30	8	1.32	8
DBSCAN-LSTM	1.37	6	1.34	7
BIRCH-LSTM	1.43	9	1.34	11
HDBSCAN-LSTM	**1.29**	15	**1.30**	13

shows that there are 15 and 13 base LSTM models f Years 2010 and 2011, respectively. That is, for each year, the dataset $D1$ is partitioned into 15 and 13 groups, respectively, for training the first-level LSTM models. The table also shows that the second-level learning by the FCC neural network effectively

TABLE V
INDIVIDUAL HDBSCAN BASED MODEL RESULTS AND THE ENSEMBLE METHOD IMPROVEMENT FOR THE SYSTEM LEVEL LOAD IN YEARS 2010 AND 2011

	ISONE (SYS) 2010	ISONE (SYS) 2011
Model	MAPE	MAPE
1	1.679	2.589
2	1.905	1.600
3	2.415	1.472
4	2.171	**1.317**
5	1.778	2.108
6	1.798	2.347
7	2.196	1.396
8	1.460	1.378
9	1.518	2.808
10	1.373	1.351
11	1.506	1.386
12	**1.307**	†9.685
13	1.448	1.377
14	1.757	-
15	2.303	-
Combined Model	**1.291**	**1.299**

further reduces the MAPE. Compared with the MAPEs in the first-level learner, the FCC achieves an average improvement in MAPE of 21.59% and 25.60% for the Year 2010 and 2011, respectively. To visualize the performance results, the forecast results of the last two weeks in 2011 predicted by the HDBSCAN based LSTM model are plotted along with the ground truth in Fig. 7. It can be seen that the forecast curve matches the ground truth tightly.

In the 2011 prediction results, the performance of model 12 is marked with a symbol "†," which indicates the worst score MAPE among all the 13 LSTM models. We carefully examine this case and plot the clustering result for this prediction in Fig. 8. It can be seen that each of the other 12 clusters has a sufficient number of samples, while only 69 samples are grouped into the 12th cluster. This level-one learner (LSTM model 12) is trained with a very small dataset. As a result, it has a comparatively weak ability of generalization. It achieves the worst performance as the features extracted by this model

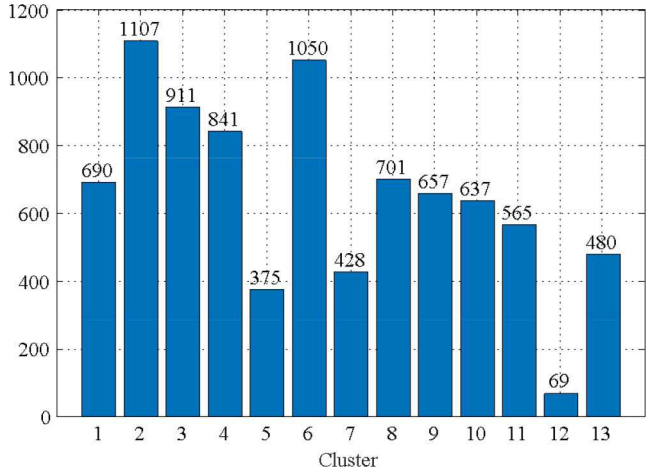


Fig. 8. Sample distribution of the HDBSCAN model for system level load prediction in Year 2011.

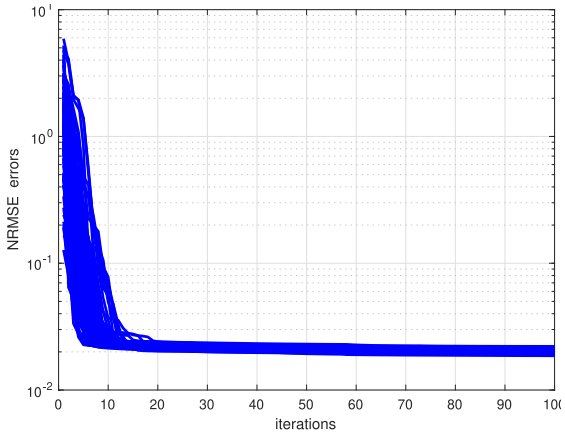
TABLE VI
THE EFFECT OF THE NUMBER OF HIDDEN NEURONS ON THE TRAINING PROCESS

Number of	ISONE (SYS) 2010 Average NRMSE		ISONE (SYS) 2011 Average NRMSE	
Hidden Neurons	Training	Testing	Training	Testing
2	0.0203	0.0212	0.0191	0.0216
3	0.0209	0.0218	0.0185	0.0209
4	0.0215	0.0223	0.0183	0.0207
5	0.0203	0.0212	0.0186	0.0209
6	0.0206	0.0214	0.0187	0.0211
7	0.0207	0.0215	0.0184	0.0206
8	**0.0192**	**0.0201**	0.0182	0.0205
9	0.0216	0.0224	0.0188	0.0212
10	0.0208	0.0217	0.0180	0.0201
11	0.0198	0.0208	**0.0179**	**0.0201**

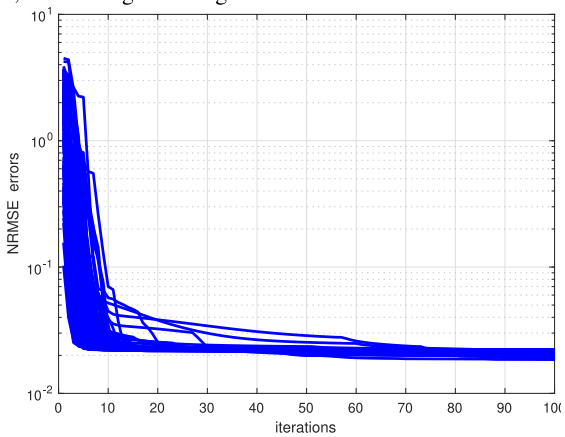
are not general enough and are only suitable and specific to the sample dataset (Cluster 12).

We further explore the effect of the number of hidden neurons in the second level of learning on the prediction. Table VI shows the average training and testing error (i.e., Normalized Root Mean Square Error) learned by the HDBSCAN based LSTM model with different numbers of hidden neurons. In each trial, the neural network with the same number of hidden neurons is trained 100 times, and the average training and testing errors are presented in the table. As shown in the table, increasing the number of hidden neurons does not guarantee to reduce the training and testing errors. The minimum training and testing errors are achieved with 8 hidden neurons for ISONE (SYS) 2010 and with 11 hidden neurons for ISONE (SYS) 2011. Finding a proper parameter (i.e., the number of hidden neurons) is vital for the training process. Thus, the grid search technique is applied in our proposed framework.

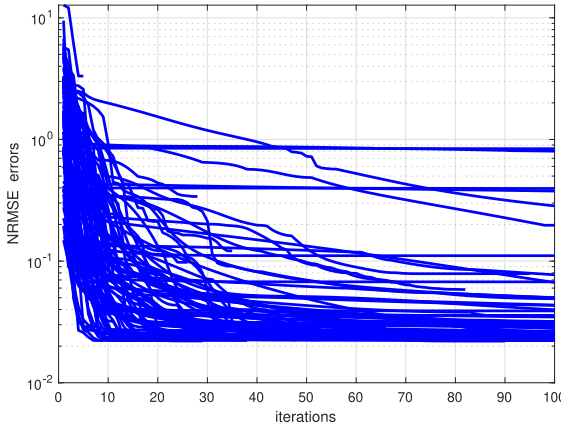
As mentioned in Section II-D, the FCC neural network's hidden neurons are activated by the tanh(·) function. In order to explain why we choose this activation function, we compare the performance (i.e., the learning curve) of different activation functions. The model is trained by tanh(·), sigmoid(·), and ReLU(·), respectively, with the same input and neural network structure. This experiment is implemented with the HDBSCAN-LSTM model, which has 3 hidden neurons, using Year 2010 data. Fig. 9 presents the learning curves, training and testing errors, as well as training and testing time. It indicates that the tanh(·) and sigmoid(·) activation functions



(a) tanh activation function: average training error (NRMSE) is 0.0210 ± 0.0011, average testing error (NRMSE) is 0.0219 ± 0.0011, and average training time is 42.0660 seconds.



(b) sigmoid activation function: average training error (NRMSE) is 0.0214 ± 0.0010, average testing error (NRMSE) is 0.0222 ± 0.0009, ad average training time is 38.1347 seconds.



(c) ReLU activation function: average training error (NRMSE) is 0.1347 ± 0.3834, average testing error (NRMSE) is 0.1331 ± 0.3698, and average training time is 42.8419 seconds.

Fig. 9. Learning curves of the ensemble neural networks (FCC) with different activation functions.

TABLE VII
COMPARISON OF THE FOUR VARIANTS OF THE PROPOSED MODEL WITH THE BASIC LSTM MODEL ON THE ISO-NE DATASET FOR WEEKLY AHEAD HOURLY LOAD FORECAST ON WEEKEND DAYS IN YEAR 2018: TESTING ERRORS

	LSTM		Kmeans++-LSTM		DBSCAN-LSTM		BRICH-LSTM		HDBSCAN-LSTM	
ZONE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ISONE(SYS)	1279.639	6.396	1172.054	5.987	1138.088	5.829	1143.666	5.878	**754.019**	**4.369**
CT	332.453	7.008	336.177	7.274	324.761	6.881	332.367	6.821	**187.77**	**4.610**
NH	108.098	5.822	107.734	5.779	105.366	5.671	108.381	5.812	**75.092**	**4.837**
ME	79.852	4.487	78.204	4.427	76.080	4.327	79.336	4.485	**60.137**	**3.680**
RI	92.787	6.411	89.279	6.566	85.895	6.243	87.864	6.606	**48.600**	**4.431**
VT	58.264	7.614	50.924	6.172	53.261	6.554	53.478	6.783	**46.746**	**5.829**
SEWASS	178.423	7.330	168.339	7.210	169.527	7.352	166.968	7.168	**113.50**	**5.512**
WCMASS	173.362	6.421	158.812	6.114	164.382	6.223	163.230	6.327	**129.867**	**5.687**
NEWASS	290.23	6.747	278.474	7.071	259.557	6.393	262.456	6.535	**177.01**	**5.059**

TABLE VIII
COMPARISON OF THE FOUR VARIANTS OF THE PROPOSED MODEL WITH THE BASIC LSTM MODEL ON THE ISO-NE DATASET FOR WEEKLY AHEAD HOURLY LOAD FORECAST ON WEEKEND DAYS IN YEAR 2018: TRAINING ERRORS

	LSTM		Kmeans++-LSTM		DBSCAN-LSTM		BRICH-LSTM		HDBSCAN-LSTM	
ZONE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ISONE(SYS)	1203.829	6.049	652.820	4.059	637.553	3.990	605.226	3.789	**576.435**	**3.412**
CT	318.441	6.335	168.854	4.373	187.350	4.745	176.095	4.699	**174.893**	**4.383**
NH	96.405	5.175	59.860	3.901	60.260	3.963	58.533	3.836	**57.924**	**3.567**
ME	59.889	3.734	47.506	3.041	45.875	2.895	44.091	2.792	**50.535**	**2.973**
RI	77.740	5.381	34.034	3.263	35.478	3.461	37.404	3.617	**36.910**	**3.274**
VT	52.537	6.644	50.396	6.019	52.812	6.403	53.261	6.554	**45.881**	**5.801**
SEWASS	147.80	5.864	94.874	4.951	103.588	5.225	92.165	5.001	**91.264**	**4.960**
WCMASS	149.845	5.502	108.455	4.609	112.473	4.824	97.647	4.406	**94.990**	**4.436**
NEWASS	142.367	3.938	108.531	3.489	116.474	3.669	94.975	3.020	**93.519**	**3.097**

TABLE IX
MAPE COMPARISON OF THE CLUSTERING BASED MODEL AND THE BASIC LSTM MODEL ON RESIDENTIAL DATA

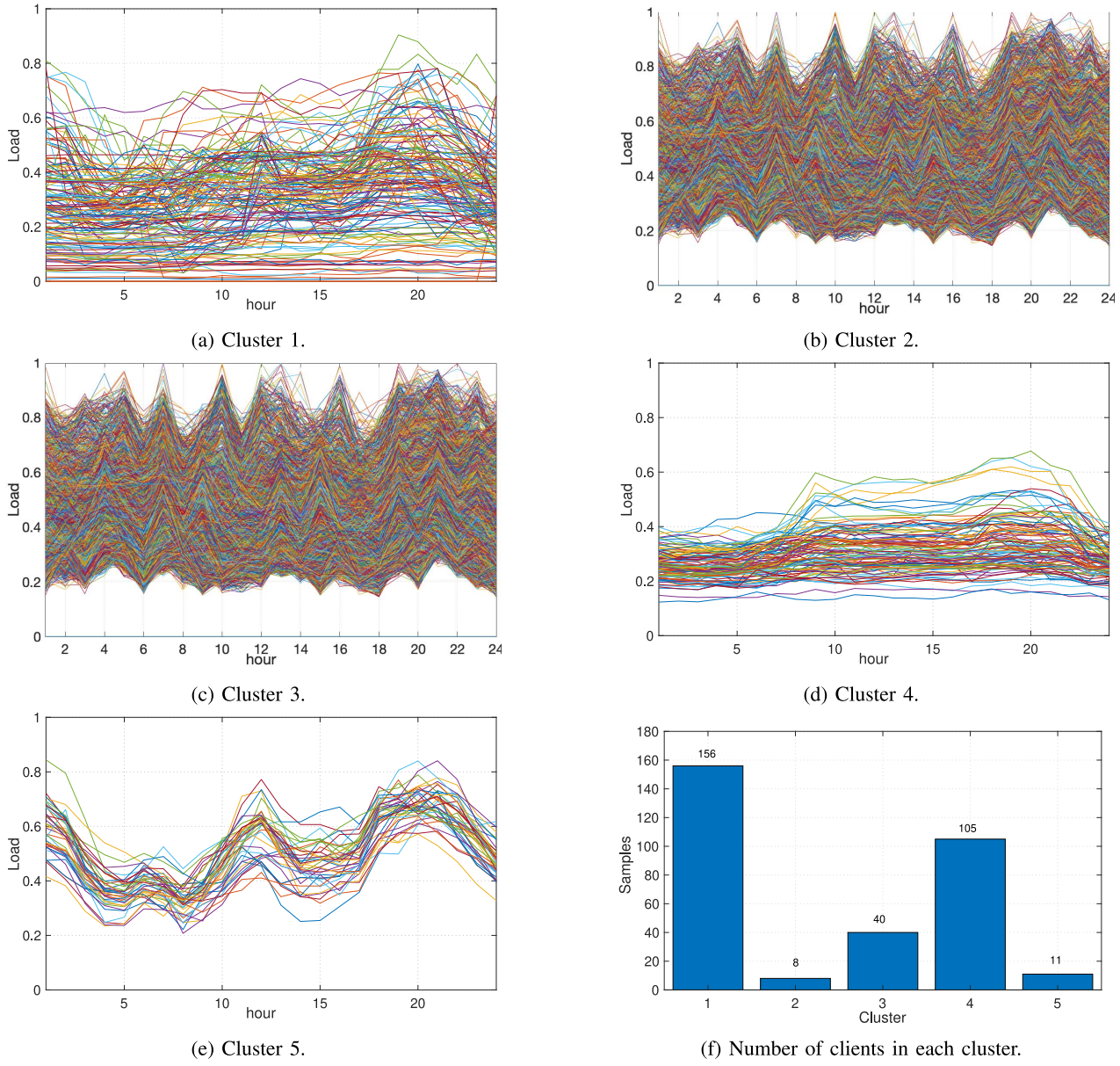
	K-means++			BIRCH			DBSCAN			HDBSCAN			LSTM		
Resident	$\tau=1$	$\tau=12$	$\tau=24$	$\tau=1$	$\tau=12$	$\tau=24$	$\tau=1$	$\tau=12$	$\tau=24$	$\tau=1$	$\tau=12$	$\tau=24$	$\tau=1$	$\tau=12$	$\tau=24$
1	**18.15**	43.59	**43.51**	18.16	55.28	46.89	23.74	45.36	44.54	19.33	**42.21**	44.32	36.15	42.88	48.85
2	3.79	7.70	8.34	3.70	7.76	8.45	**3.66**	**6.78**	8.33	3.75	7.33	**7.96**	14.51	10.27	9.78
3	11.90	17.06	15.81	15.54	**15.15**	15.24	12.72	15.4 1	16.33	**11.21**	15.24	**15.12**	22.23	24.50	23.12
4	9.31	16.41	13.92	**8.53**	**10.19**	**12.90**	12.09	12.29	13.11	8.83	12.55	12.94	36.53	38.55	38.34
5	12.63	16.27	14.76	14.53	20.52	**14.40**	12.57	15.71	14.90	**11.24**	**14.91**	15.22	27.74	23.58	24.19

are more stable than the ReLU(·) function. Although the sigmoid(·) function takes less time for training, the tanh(·) function achieves a slightly better performance on reducing the training and testing error.

The *second case* we examine is to forecast week-ahead power load on weekends (i.e., for Saturday and Sunday) at both the zone level and system level for the Year 2018. The data from 2015 to 2017 are used for training the models. The output is the weekend's hourly load values. In this task, we only use historical temperature data as a feature. The current temperature (i.e., at $t+\tau$) is not used in this forecast, which is different from the previous case. This is because in practice, weekly ahead weather forecast is not as precise as day-ahead weather forecast. In order to mimic the actual situation in forecasting, we only use the feature information that is available at the forecasting time instance in this study (i.e., no future information is available). Therefore, the input of this case is weekly lagged temperature and power load time series data.

The evaluation results of both Root Mean Square Error (RMSE) and MAPE are summarized in Table VII (testing errors) and Table VIII (training errors), for both the overall system-level load prediction (the first row) and that for each of the zones in the New England area (the remaining eight rows). We compare the four variants of the proposed framework with the basic LSTM model using the same input. Apparently, our proposed framework performances better than the traditional LSTM model. The HDBSCAN based model consistently outperforms all the other models in this experiment.

2) Residential Level Prediction Performance: We next study the load forecasting problem for individual clients using the proposed ensemble learning model on the Residential Electricity Consumption dataset [39]. The electricity load data is aggregated from every 15 minutes to one hour. The aggregated dataset is spitted into three parts as described in Section IV-A2. Then the 320 clients are classified into several groups using the HDBSCAN clustering algorithm based on



(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

(f) Number of clients in each cluster.

Fig. 10. Distribution of classified residents based on the DBSCAN clustering algorithm.

the data of the first three months in 2012. Fig. 10 presents the clustering result. It shows that the consumers are grouped into five clusters. In Figs. 10(a)-(e), each curve represents the normalized load of a client, while Fig. 10(f) shows the number of clients in each cluster. We find that each cluster is visually different. For example, the load curves in Cluster 4 are all relatively flat and are close to 0.5, the load curves in Clusters 2 and 3 are serrated and ranging from 0.2 to 1, and the load curves in Cluster 5 exhibit an obvious daily pattern (indicating these are residential clients).

In each cluster, we randomly select a client to predict its load. Thus, for each client, historical load time series right before time step t is used as the training set, where the window size W is set to 168 (the number of hours in seven days). The dimension of input $m \times W$ at time t is 1×168 since we only use the historical load data (i.e., $m = 1$). The output is the predicted load for a future time $t + \tau$.

We use the traditional LSTM model as a baseline scheme. Table IX summarizes the evaluation results in the form of MAPE for the five chosen clients (one from each cluster as shown in Fig. 10). The horizon τ is set to be 1, 12, and 24, respectively, which means we predict hour ahead, half-day ahead, and day-ahead load values for the five selected clients. Compared with LSTM, the proposed ensemble learning models achieve a much higher precision in this experiment. For example, for Client 4, the BIRCH MAPEs are 23.35%, 26.43%, and 33.64% of the corresponding LSTM MAPEs for $\tau = 1$, $\tau = 12$, and $\tau = 24$, respectively. The best result for different clients and horizon τ is different. However, the proposed ensemble learning models all achieve the best performance. Among all the results, BIRCH and HDBSCAN

based models perform better, which achieve the lowest errors comparing to others. Considering HDBSCAN is an improved algorithm of DBSCAN, density-based algorithm HDBSCAN and hierarchical algorithm BIRCH are superior to partitioning algorithm K-means++, which suffers from outliers or noise, in this case.

For all models, the MAPE of Client 1 is relatively high, while the MAPE of all the other clients are all below 21. From Fig. 10, we can see that in cluster one, there is no obvious trend for this group of data, which might explain why this group of data is difficult to forecast. It is extremely challenging to accurately predict every client's load due to different lifestyles or activities. Classifying the clients and predict load by the group is quite feasible as shown in this experiment.

V. Conclusion

In this paper, we proposed a novel ensemble learning approach based on deep learning (i.e., LSTM) and unsupervised learning (i.e., clustering) for load forecasting. In the first level of learning, a set of LSTM models are generated by data clusters. In the second-level learner, an FCC neural network enhanced by a modified second-order optimization algorithm fuses and improves the predictions by the first-level learners. Superior performance was demonstrated by using two real-world datasets for load forecasting at both the system and client levels. Such accurate predictions can be very helpful for energy management in the urban grid system.

References

[1] L. Wang, S. Mao, and B. Wilamowski, "Short-term load forecasting with LSTM based ensemble learning," in *Proc. IEEE Green Comput. Commun. (GreenCom)*, Atlanta, GA, USA, Aug. 2019, pp. 793–800.

[2] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, pp. 1–55, Sep. 2014.

[3] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Netw. Appl. J.*, vol. 19, no. 2, pp. 171–209, Apr. 2014.

[4] *Introduction to Sustainable Urban Energy Systems*, Asia–Pac. Urban Energy Assoc., Bangkok, Thailand, 2019. [Online]. Available: https://www.apuea.org/index.php/urban-energy-systems/introduction-to-urban-energy-systems2

[5] H. Ghazzai and A. Kadri, "Joint demand-side management in smart grid for green collaborative mobile operators under dynamic pricing and fairness setup," *IEEE Trans. Green Commun. Netw.*, vol. 1, no. 1, pp. 74–88, Mar. 2017.

[6] F. Uddin, "Energy-aware optimal data aggregation in smart grid wireless communication networks," *IEEE Trans. Green Commun. Netw.*, vol. 1, no. 3, pp. 358–371, Sep. 2017.

[7] K. Wang, H. Li, S. Maharjan, Y. Zhang, and S. Guo, "Green energy scheduling for demand side management in the smart grid," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 2, pp. 596–611, Jun. 2018.

[8] M. Collotta and G. Pau, "An innovative approach for forecasting of energy requirements to improve a smart home management system based on BLE," *IEEE Trans. Green Commun. Netw.*, vol. 1, no. 1, pp. 112–120, Mar. 2017.

[9] Y. Wang, G. Cao, S. Mao, and R. Nelms, "Analysis of solar generation and weather data in smart grid with simultaneous inference of nonlinear time series," in *Proc. IEEE INFOCOM Workshop SmartCity*, Hong Kong, Apr. 2015, pp. 672–677.

[10] Y. Wang, Y. Shen, S. Mao, G. Cao, and R. Nelms, "Adaptive learning hybrid model for solar intensity forecasting," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1635–1645, Apr. 2018.

[11] Y. Wang, Y. Shen, S. Mao, X. Chen, and H. Zou, "LASSO and LSTM integrated temporal model for short-term solar intensity forecasting," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2933–2944, Apr. 2019.

[12] T. Strasser *et al.*, "A review of architectures and concepts for intelligence in future electric energy systems," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2424–2438, Apr. 2015.

[13] 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.

[14] 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.

[15] D. L. Marino, K. Amarasinghe, and M. Manic, "Building energy load forecasting using deep neural networks," in *Proc. Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Florence, Italy, Oct. 2016, pp. 7046–7051.

[16] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.

[17] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, and J. He, "Short-term load forecasting with deep residual networks," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3943–3952, Jul. 2018.

[18] Y. Wang, N. Zhang, Y. Tan, T. Hong, D. S. Kirschen, and C. Kang, "Combining probabilistic load forecasts," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3664–3674, Jul. 2018.

[19] W. Zhang, H. Quan, and D. Srinivasan, "An improved quantile regression neural network for probabilistic load forecasting," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 4425–4434, Jul. 2019.

[20] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: Chapman & Hall, 2012.

[21] P. Smyth and D. Wolpert, "Linearly combining density estimators via stacking," *Mach. Learn.*, vol. 36, nos. 1–2, pp. 59–83, Jul. 1999.

[22] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Burlington, MA, USA: Morgan Kaufmann, 2011.

[23] C. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[24] D. Arthur and S. Vassilvitskii, "*k*-means++: The advantages of careful seeding," in *Proc. ACM/SIAM Symp. Discrete Alg. (SODA)*, New Orleans, LA, USA, Jan. 2007, pp. 1027–1035.

[25] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, Jun. 1996.

[26] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discovery Data Mining (KDD)*, Portland, OR, USA, Aug. 1996, pp. 226–231.

[27] L. McInnes, J. Healy, and S. Astels, "HDBSCAN: Hierarchical density based clustering," *J. Open Source Softw.*, vol. 2, no. 11, pp. 205–206, 2017.

[28] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, New Orleans, LA, USA, Nov. 2017, pp. 33–42.

[29] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl.Discovery Data*, vol. 10, no. 1, Jul. 2015, Art. no. 5.

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

[31] D. Hunter, H. Yu, M. S. Pukish, III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures–a comparative study," *IEEE Trans. Ind. Informat.*, vol. 8, no. 2, pp. 228–240, May 2012.

[32] X. Wang, X. Wang, and S. Mao, "ResLoc: Deep residual sharing learning for indoor localization with CSI tensors," in *Proc. IEEE Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, Montreal, QC, Canada, Oct. 2017, pp. 1–6.

[33] J. Martens, "Deep learning via Hessian-free optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Haifa, Israel, Jun. 2010, pp. 735–742.

[34] J. E. Vitela and J. Reifman, "Premature saturation in backpropagation networks: Mechanism and necessary conditions," *Neural Netw.*, vol. 10, no. 4, pp. 721–735, Jun. 1997.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2017. [Online]. Available: arXiv:1412.6980.

[36] J. Fan, "Accelerating the modified Levenberg–Marquardt method for nonlinear equations," *Mathe. Comput.*, vol. 83, no. 287, pp. 1173–1187, May 2014.

[37] J. Fan and J. Pan, "Convergence properties of a self-adaptive Levenberg–Marquardt algorithm under local error bound condition," *Comput. Opt. Appl.*, vol. 34, no. 1, pp. 47–62, May 2006.

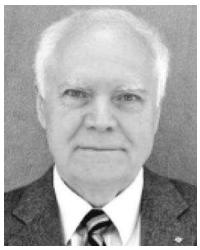
[38] *ISO New England Zonal Information*, ISO New England, Holyoke, MA, USA, 2019. [Online]. Available: https://iso-ne.com/isoexpress/web/reports/pricing/-/tree/zone-info
[39] *Electricity Load Diagrams 2011–2014 Data Set*, UC Irvine Mach. Learn. Repository, Irvine, CA, USA, 2019. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[40] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Mining Knowl. Discovery*, vol. 7, no. 4, pp. 349–371, Oct. 2003.
[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Mar. 2015. [Online]. Available: arXiv:1502.03167.
[42] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 3, pp. 1464–1468, Jun. 1997.

Lingxiao Wang received the B.E. degree in electrical engineering and automation from Nanjing University of Information Science and Technology, Nanjing, China, in 2012, and the M.S. degree in electrical and computer engineering from Auburn University, Auburn, AL, USA, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His research interests include deep learning, neural network optimization, and time-series prediction.

Shiwen Mao (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from Polytechnic University (currently, New York University Tandon School of Engineering), Brooklyn, NY, USA.

He held the McWane Professorship from 2012 to 2015. He is currently the Samuel Ginn Endowed Professor with the Department of Electrical and Computer Engineering and the Director of the Wireless Engineering Research and Education Center, Auburn University, Auburn, AL, USA. His research interests include wireless networks, multimedia communications, and smart grid.

Dr. Mao received the IEEE ComSoc TC-CSR Distinguished Technical Achievement Award in 2019, the IEEE ComSoc MMTC Distinguished Service Award in 2019, the Auburn University Creative Research and Scholarship Award in 2018, the 2017 IEEE ComSoc ITC Outstanding Service Award, the 2015 IEEE ComSoc TC-CSR Distinguished Service Award, and the 2013 IEEE ComSoc MMTC Outstanding Leadership Award, and the NSF CAREER Award in 2010. He was a co-recipient of the 2018 IEEE ComSoc MMTC Best Journal Paper Award, the 2017 IEEE ComSoc MMTC Best Conference Paper Award, the Best Demo Award from IEEE SECON 2017, the Best Paper Awards from IEEE GLOBECOM 2019, 2016, and 2015, IEEE WCNC 2015, and IEEE ICC 2013, and the 2004 IEEE Communications Society Leonard G. Abraham Prize in the field of communications systems. He is a Distinguished Speaker and was a Distinguished Lecturer from 2014 to 2018 of the IEEE Vehicular Technology Society. He is an Area Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, the IEEE INTERNET OF THINGS JOURNAL, IEEE/CIC CHINA COMMUNICATIONS, and *ACM GetMobile*, and an Associate Editor of the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, the IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE MULTIMEDIA, IEEE NETWORKING LETTERS, *Ad Hoc Networks* (Elsevier), and *Digital Communications and Networks* (Elsevier). He is a member of ACM.



Bogdan M. Wilamowski (Life Fellow, IEEE) received the M.S. degree in computer engineering, the Ph.D. degree in neural computing, and the Habilitation degree in integrated circuit design from Gdansk University of Technology, Gdańsk, Poland, in 1966, 1970, and 1977, respectively. He was the Director of the Alabama Micro/Nano Science and Technology Center, Auburn University, Auburn, AL, USA, from 2003 to 2016, where he is currently a Professor Emeritus with the Department of Electrical and Computer Engineering. He is also with the University of Information Technology and Management, Rzeszów, Poland. He served as the Vice President of the IEEE Computational Intelligence Society from 2000 to 2004, the President of the IEEE Industrial Electronics Society from 2004 to 2005, and a member of the IEEE Board of Directors from 2012 to 2014. He also served as an associate editor for numerous other journals. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS from 2007 to 2010 and the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS from 2011 to 2013.

R. M. Nelms (Fellow, IEEE) received the B.E.E. and M.S. degrees in electrical engineering from Auburn University, Auburn, AL, USA, in 1980 and 1982, respectively, and the Ph.D. degree in electrical engineering from Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 1987. He is currently a Professor and the Chair of the Department of Electrical and Computer Engineering, Auburn University. His research interests are in power electronics, power systems, and electric machinery. In 2004, he was named an IEEE Fellow for technical leadership and contributions to applied power electronics. He is a Registered Professional Engineer in Alabama.