Deep Learning for Household Load Forecasting—A Novel Pooling Deep RNN

Heng Shi, Minghao Xu, and Ran Li, *Member, IEEE*

***Abstract*—The key challenge for household load forecasting lies in the high volatility and uncertainty of load profiles. Traditional methods tend to avoid such uncertainty by load aggregation (to offset uncertainties), customer classification (to cluster uncertainties) and spectral analysis (to filter out uncertainties). This paper, for the first time, aims to directly learn the uncertainty by applying a new breed of machine learning algorithms—deep learning. However, simply adding layers in neural networks will cap the forecasting performance due to the occurrence of over-fitting. A novel pooling-based deep recurrent neural network is proposed in this paper which batches a group of customers' load profiles into a pool of inputs. Essentially the model could address the over-fitting issue by increasing data diversity and volume. This paper reports the first attempts to develop a bespoke deep learning application for household load forecasting and achieved preliminary success. The developed method is implemented on Tensorflow deep learning platform and tested on 920 smart metered customers from Ireland. Compared with the state-of-the-art techniques in household load forecasting, the proposed method outperforms ARIMA by 19.5%, SVR by 13.1% and classical deep RNN by 6.5% in terms of RMSE.**

***Index Terms*—Big data, deep learning, load forecasting, long short-term memory, machine learning, neural network, smart meter.**

I. Introduction

DEMAND side response (DSR) plays a key component in achieving the political goals set in the U.K. and EU energy sector [1], [2]. The popularisation of smart meters will make the DSR easier than ever for domestic customers [1]. Various direct and indirect control methods have been proposed to realise DSR [3]–[5] given that household load can be accurately forecasted.

Extensive and comprehensive review papers on point load forecasting at aggregated level already exist [14]–[26], [40]–[42]. However, the literature on individual household load forecasting is actually limited [5]–[14] as it is widely acknowledged that short-term load forecasting (STLF) at such granular level is extremely challenging due to significant uncertainty and volatility [6]–[8] underlying the smart metering data.

Manuscript received August 7, 2016; revised December 20, 2016; accepted March 10, 2017. Date of publication March 22, 2017; date of current version August 21, 2018. Paper no. TSG-01045-2016. *(Corresponding author: Ran Li.)*

The authors are with the Department of Electronic and Electrical Engineering, University of Bath, Bath BA2 7AY, U.K. (e-mail: h.shi@bath.ac.uk; m.xu2@bath.ac.uk; r.li2@bath.ac.uk).

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.2017.2686012

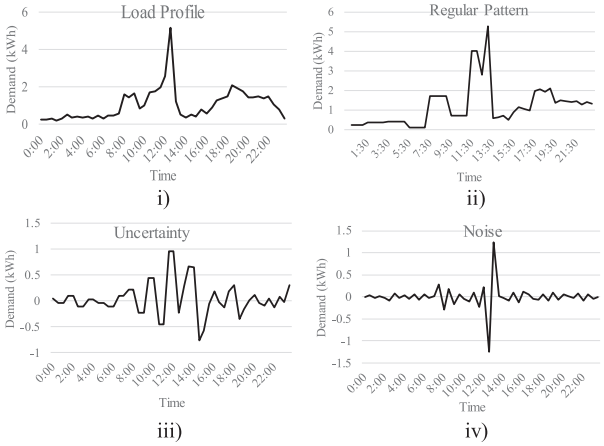


Fig. 1. Sketch of load composition: i) original load, ii) regular pattern, iii) uncertainty and iii) noise.

Experiments have been carried out by [6], [7], and [9]–[13] to benchmark state-of-art methods for STLF at individual household level. Testing methods include time-series analysis (e.g., ARIMA and exponential smoothing) and machine learning approaches (e.g., neural networks and support vector machine). Similar findings are reported in both papers [9], [10] as none of the classical methods could beat linear regression or even simple persistence forecasting (i.e., tomorrow equals today) at individual household level.

A. Uncertainty

The complexity of household load forecasting lies in the significant volatility and uncertainty. In the context of STLF, load could be decomposed into three components. As shown in Fig. 1, the original household load profile i) is decomposed into: ii) regular pattern, which reflects the periodical load inherited from historical data; iii) uncertainty, which is the aperiodic part influenced by external factors such as weather, events and customer behaviour and iv) noise, the residue load which cannot be physically explained [14], [15].

Most forecasting models focus on the regular pattern as it is more predictable and makes up a dominating proportion at the aggregated level. However, household demand is composed of a substantially larger share of uncertainty. At this level, uncertainty is more influenced by customer behaviour, which is too stochastic to predict. Therefore, the nature of the challenge is to forecast load with significant uncertainty.

To tackle this problem, three categories of methods have been reported:

1) Using clustering/classification techniques to group similar customers, days or weather [6], [16]–[19] in the hope of reducing the variance of uncertainty within each cluster. However, the performance is heavily dependent on the data.
2) Using aggregated smart metering data to cancel out the uncertainties [20]–[23] so that the aggregated load exhibits mostly regular patterns and easier to predict, yet the prediction is obviously only at aggregated level.
3) Using pre-processing techniques, mostly spectral analysis such as wavelet analysis [24], Fourier transforms [25] and empirical mode decomposition (EMD) [26] aiming to separate the regular patterns from the other two components. This method can be ruled out in household load forecasting due to its significantly lower proportion of regular patterns.

To the best of our knowledge, the existing methods towards the problem are indirect, which aim to avoid uncertainty by reducing (clustering) or canceling out (aggregation) or separating (spectral analysis) the uncertainty. This paper aims to explore the possibility of deploying the state-of-art deep learning algorithm to directly learn uncertainties in their raw forms. Deep learning is a branch of machine learning methods relying on 'deep' architectures, which are compositions of multiple processing layers in the neural network, enabling the learning of highly non-linear, complicated relationships and correlations that are beyond the reach of traditional shallow architectures. Deep learning has achieved many breakthroughs in tackling sophisticated problems and becomes the most promising technique in data science community, for example, Google Goggles, Alpha Go [27] and new drugs design [28]. Attempts have been reported in [29] and [30] to adopt deep learning for time series forecasting.

B. Overfitting

However, direct implementation of deep learning in household load forecasting will not necessarily provide significant improvement. A preliminary test has been carried out by the authors to benchmark the performance of household load forecasting by a neural network with a different number of layers.

The indicative result shown in Fig. 2 demonstrates the occurrence of overfitting when the number of layers reaches 3. As the number of layers increases, the forecasting error decreases before 3 layers. However, further increase of the network depth will see a rebound of error. As acknowledged in [31], the primary drivers are model capacity and training epochs (training iterations). To prevent excessive training iterations, we implemented early stopping technique to find optimal number of training iterations. In detail, dataset is split into training, validation and test sets. In each of the training iteration, the process will stop if the RMSE on validation set no longer decreases for a certain number of epochs.

Model capacity refers to the ability to fit a wide variety of functions. Model with large capacity tends to suffer from overfitting. To avoid excessive model capacity, one way is to increase the data diversity so that sufficient model capacity is becoming an advantage rather than a burden. Particularly for Deep Learning techniques, whose model capacity is much larger than the rest of models. When increasing the deep neural network layer number, the inherent parameters with the network will grow exponentially and eventually become excessive for the available training data. As a result, the model will begin to capture the noise and fit the training data too well, hence impact the predictive performance in a negative way.

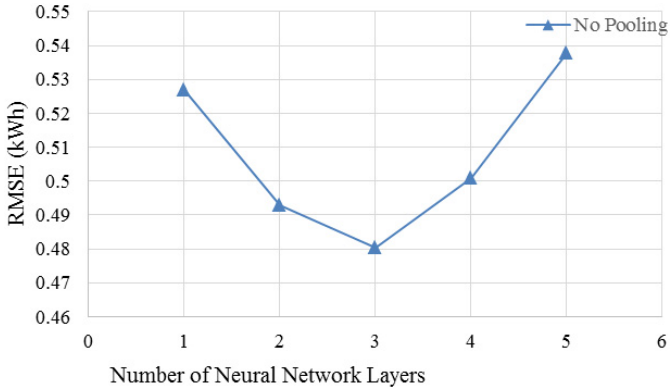


Fig. 2. Household load forecasting performance by neural networks from shallow to deep.

In order to enable the power of deep learning algorithm in our problem, a novel pooling-based deep recurrent neural network (PDRNN) is proposed. The pooling technique will batch customers and input into the deep recurrent network as a whole.

The key contributions of this paper are as follows:

1) New technique: this paper for the first time explores the feasibility of a cutting edge algorithm, deep learning, in the application of load forecasting at individual household level.
2) New problem: although deep learning has received high expectation in forecasting community, our experiment indicates that deep learning is more prone to over-fitting compared with its 1980s' cousin, neural networks. This is possibly due to more parameters and relatively fewer data.
3) New method: we propose a novel pooling method to address the over-fitting issue by introducing a new data dimension: historical load data of neighbours. The idea is to use the interconnected spatial information to compensate insufficient temporal information. The proposed load profile pool allows for the correlations and interactions between neighbouring households. New features can be automatically generated through deep layers hierarchically and thus increases the inputs volume and diversity.

The rest of the paper is organised as follows: Section II briefly introduces the rationale for applying Deep Learning in household STLF tasks and the specific LSTM technique applied in the paper. Section III proposes pooling strategy and pooling-based DRNN method. Section IV explains

the implementation process on GPU-based high-performance computing platform, as well as the details of experiment setup. In Section V, results are demonstrated through comparison with previous state-of-the-art methods (ARIMA, SVR), shallow learning (normal RNN), classical deep learning (DRNN) and proposed deep learning (PDRNN). Conclusions are drawn in part VI.

II. Deep Learning

Deep learning is a branch of machine learning methods lying on 'deep' network architectures. The concept of 'deep learning' has been proposed for decades with the name 'cybernetics' in 1943, by McCulloch and Pitts [32]. However, it has been regarded as being more of a fancy concept than an applicable technology, due to three major technical constraints. The three technical constraints are: 1) lack of sufficient data, 2) lack of computing resources for large network size, and 3) lack of efficient training algorithm.

Recently, the constraints are tackled by the digitalization of modern society and the development of high-performance computing. Furthermore, Hinton *et al.* [33] made a breakthrough in efficient deep neural network training via a strategy called greedy layer-wise pre-training, which enables practical implementations of deep learning.

Deep learning has recently seen phenomenal success in various areas including: 1) Computer Vision (CV) such as Google Goggles, which uses deep learning for object recognition; 2) expert systems such as Alpha Go designed by DeepMind [27] and 3) medical sciences, which employs deep learning to assist pharmaceutical companies in new drugs design [28].

A. Rationale of Using Deep Learning

Deep learning is regarded as one the most promising techniques to this study due to two superior attributes compared with "shallow" architecture.

1) To Learn Highly Non-Linear Relationships: In the problem of STLF at the household level, the inherent uncertainties are caused by differing known or unknown external factors simultaneously. These factors, ranging from weather conditions, temperatures to property size, photovoltaic generations are correlated to each other, which leave a highly non-linear impact to the household load. For example, temperature and sunshine duration are two of the external factors which are highly correlated to each other, i.e., the increase in sunshine duration can result in higher temperature in the region.

The essence of neural networks and other load forecasting methods is to learn the non-linear relationships between feed-in inputs and outputs by constructing linear or non-linear functions that approximate the real relationships between inputs and outputs. The universe approximation theorem [34] indicates the neural networks can make accurate approximations towards any non-linear functions with sufficient network size. The approximation capability of a shallow network is much lower than that of a deep network even with extra neurons at each layer. The reason is that, in 'shallow' neural networks, hidden neurons are learning the non-linear combinations of inputs as the features. However, 'deep' neural network can learn the non-linear combinations of features in deeper layers of the network, hence naturally learns the highly non-linear correlations.

2) To Learn Shared Uncertainties: The uncertainties are normally coming from external sources which make consistent impacts on differing households. Therefore, these uncertainties can be commonly shared within a group of customers at similar locations and time. However, these uncertainties are not always evenly shared among households. For example, the temperature increase can impact most of the households within a region, while the increase in sunshine duration mainly affects households with PV installed.

In 'deep' architecture, one of the most exciting properties is that it can learn load features hierarchically. Features with different sharing levels will be learned at different layers. Load features learned in higher layers are normally the combination of lower layer features. With respect to former example, the temperature rise features are normally learned at a lower level, since it can be concluded directly from inputs. However, the impact from sunshine hour is influenced by features like temperature, PV installation, and household direction, and hence should be learned at higher network layers. With this property, deep learning is exceptional for learning multiple uncertainties with differing sharing levels in household load.

B. Deep RNN With Long Short-Term Memory Unit

Typical architecture designs of deep learning including, Convolutional Deep Neural Networks (CNN), Deep Sparse Autoencoder (DSA), Deep Recurrent Neural Networks (DRNN), Multi-Layer Perceptions (MLP), Deep Restricted Boltzmann Machines (DRBM), etc. [31]. As a state-of-the-art deep learning architecture specifically designed for time-series forecasting, DRNN is employed to perform STLF for households in this paper.

The architecture of deep-RNN is stacking multiple RNN layers together into a 'deep' architecture. Most successful implementation of Deep-RNN is in the area of Speech Recognition [35], which is also one-dimensional time-series data with high uncertainty. In terms of the specific implementation of RNN layers, a state-of-the-art RNN, named Long Short-Term Memory (LSTM) has been employed to approach the best performance of RNN.

In this section, the deep-RNN architecture is introduced firstly, and then the implementation of LSTM units are followed.

1) Deep Recurrent Neural Network (Deep-RNNs): In deep-RNNs, the sharing states are decomposed into multiple layers in order to gain nice properties from 'deep' architectures. Experimental evidence has been given by [35] and [36] to suggest the significant benefit of building RNNs with 'deep' architectures.

The computational graph and its unfolding topological graph is presented in Fig. 3 to demonstrate the working process of a deep-RNN with N layers.

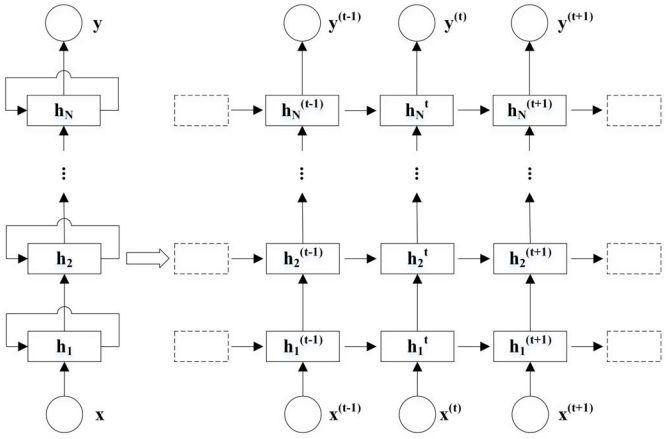


Fig. 3. The computational graph and unfolded topological graph of an N layer deep-RNN.

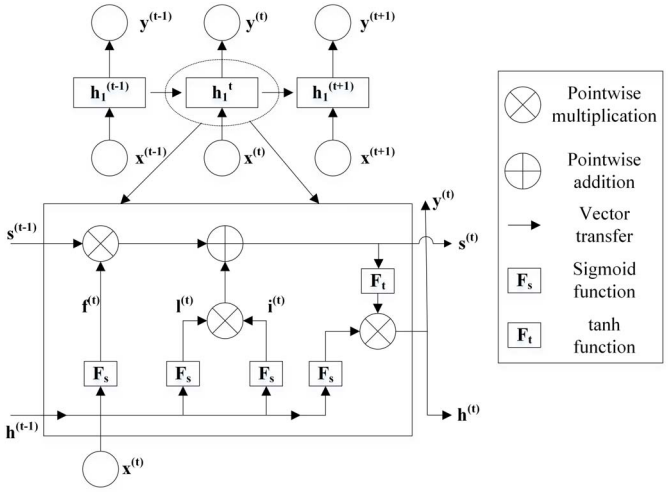


Fig. 4. The computational graph and unfolded topological graph.

In the computational graph, the RNN aims to map the input sequence of x values into corresponding sequential outputs: y. As presented in computational graph, the learning process conducted every single time step from $t = 1$ to $t = \tau$. For time step t, the network neuron parameters at l^{th} layer update its sharing states with following equations [31]:

$$a_1{}^{(t)} = b_1 + W_1 \cdot h_1{}^{(t-1)} + U_1 \cdot x^{(t)} \tag{1}$$

$$h_l{}^{(t)} = activation_{function(a_l{}^{(t)})} \quad for\ l = 1, 2, .., N \tag{2}$$

$$a_l{}^{(t)} = b_l + W_l \cdot h_l{}^{(t-1)} + U_l \cdot h_{l-1}{}^{(t)} \quad for\ l = 2, 3, \ldots, N \tag{3}$$

$$y^{(t)} = b_N + W_N \cdot h_N{}^{(t-1)} + U_N \cdot h_N{}^{(t)} \tag{4}$$

$$L = loss\_function\left(y^{(t)}, y_{target}{}^{(t)}\right) \tag{5}$$

where $x^{(t)}$ is the data input at t^{th} time step, $y^{(t)}$ is the corresponding prediction, and $y_{target}{}^{(t)}$ is the true values of output targets. $h_l{}^{(t)}$ is the sharing states of l^{th} network layer at time step t. $a_l{}^{(t)}$ represents the input value of l^{th} layer at time step t, which consists of three components: 1) t^{th} time step input $x^{(t)}$ or sharing state $h_{l-1}{}^{(t)}$ at time t from $l-1^{th}$ layer, 2) bias b, and 3) sharing states $h_l{}^{(t-1)}$ at current network layer l from last time step $t-1$. Due to the sharing properties of RNNs, the algorithm is thus capable to learn uncertainties repeated in previous time steps.

2) Boosting With Long Short-Term Memory (LSTM) Unit: Long short-term memory unit refers to a specific architecture of RNNs, which aims to tackle long-term dependencies challenge unsolved in earlier RNN architectures. When learning time-series data, RNNs aim to learn representations of patterns repeatedly occurred in the past, by sharing parameters across all time steps. However, the memory of past learned patterns can fade as time goes on. In the figure, the dependencies of earliest two inputs $x^{(0)}$ and $x^{(1)}$ becomes weak in output $y^{(t)}$ when it is reasonably large.

LSTM is hence designed to tackle this challenge by creating paths where the gradient can flow for long durations. In order to demonstrate how LSTM can memorize long-term patterns, the computational graph of LSTM is illustrated in following Fig. 4.

Fig. 4 presents a typical LSTM cell. Apart from traditional RNN units, LSTM cells have a special sharing parameter vector called memory parameter vector $s^{(t)}$ and are deployed to keep the memorized information. In each of the time steps, the memory parameter has three operations: 1) discard useless information from memory vector $s^{(t)}$; 2) add new information $i^{(t)}$ selected from input the $x^{(t)}$ and previous sharing parameter vector $h^{(t-1)}$ into memory vector $s^{(t)}$; 3) decide new sharing parameter vector $h^{(t)}$ from memory vector $s^{(t)}$.

As shown in the LSTM cell, the sharing memory parameters $h^{(t)}$ are passing through differing time steps only with two operations to memorize new information and forget out-of-time memories. Therefore, the sharing memory can keep useful information for a fairly long time and result in RNN performance enhancement.

III. Proposed Methodology

In this section, the proposed PDRNN is presented for STLF at the household level. Details of this methodology are illustrated in Fig. 5.

In general, the proposed method consists of two stages: 1) load profiles pooling, and 2) household STLF with deep-RNN. The detailed rationale and design of each stage are further discussed in the following sub-sections.

A. Stage 1: Load Profiles Pooling

In the 1^{st} stage, households' load profiles are batched into a load profile pool. The pool is fed into the 2^{nd} stage as input so that forecasting is not only based on targeted household's own data, but also load profiles of his neighbours in the pool.

1) Rationale of Pooling Strategy: The pooling strategy is designed to tackle the two major challenges of STLF at the household level, i.e., the overfitting issue and the inherent high uncertainties in household load profiles.

The overfitting issue is one of the technical constraints when applying deep learning in load forecasting. Because

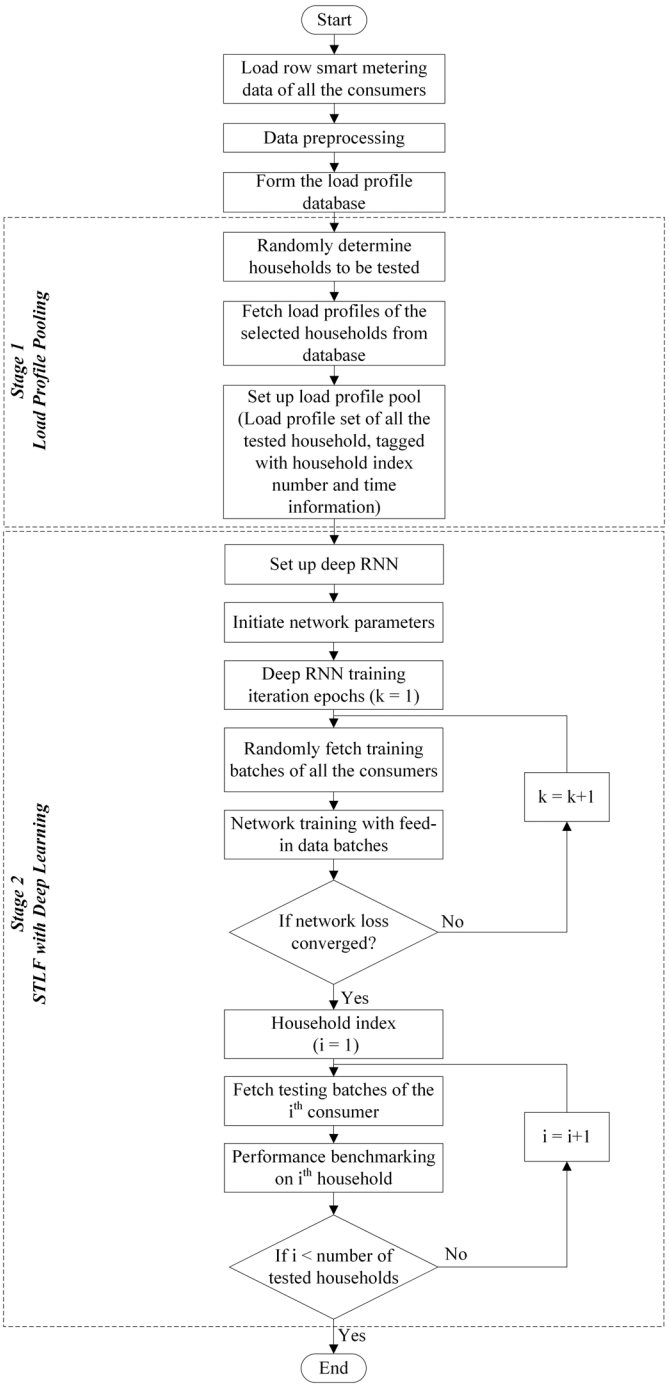


Fig. 5. Flowchart of proposed two-stage STLF methodology.

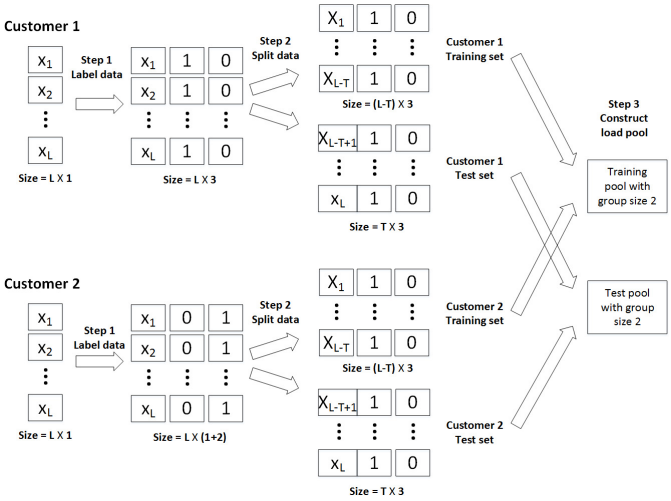


Fig. 6. Example of load pool construction with 2 customers group.

of the inherent large amount of neural layers in deep learning networks, the available historical load profile data in households are normally insufficient, which even can cause grave overfitting with a fairly small amount of network layers. The pooling stage can increase the data volume for load forecasting, which hence delays the presence of overfitting.

Because of the contingency of the load data, the inherent load uncertainties are extremely hard to be learned or modeled. However, some of the uncertainties are caused by common external factors, such as weather conditions, the day of the week, etc. Their effects are normally sharing across many customers. According to the information theory, the data diversity represents the amount of information contained. Therefore, sufficient diversity in customer load is the prerequisite for learning these common sharing uncertainties. In this stage, pooling customers' load profiles together is basically to increase the diversity in load dataset, hence increases the information related to common sharing uncertainties. Consequently, it enables the deep recurrent network to perform more accurate load forecasting by understanding these common sharing uncertainties.

2) Design of Pooling Strategy: In this paper, the household load profiles are captured from smart meters half-hourly. Therefore, the daily load profiles are of the form of 48-data-point values. Due to time connectivity of household load between continuous dates, the load samplings on d^{th} date are continuous with $(d-1)^{th}$ and $(d+1)^{th}$ dates. In order to keep this property in data, the load profile pool uses a long vector sequence, consisting of concatenated load profiles of multiple continuous dates starting from the first date of historical data. The denotation is:

$$\overrightarrow{X^{(c)}} = \left(\overrightarrow{X^{(c)}}_1, \overrightarrow{X^{(c,2)}}_2, \ldots, \overrightarrow{X^{(c,L)}}_L\right) \tag{6}$$

where L represents the total length of the demand sequence data for c^{th} *customers*. The load profile pool is then generated through 3 steps: 1) add customer id label in the form of dummy variables, 2) split data into training and test sets, 3) merge all training data to construct training pool, then construct test pool with the same process. In order to clarify the process of pool construction, a simplified illustration of 2 customers pool is presented in Fig. 6.

As illustrated above, 1^{st} and 2^{nd} customers' demand are noted as two data sequences with size $L \times 1$ and $L \times 1$. In the first step, the demand data will be labelled with dummy variables to identify its customer id. In the example, the demand data are expanded with size $L \times 3$ and $L \times 3$. The number of expanded columns is equivalent to customer group size. In the second step, demand data of each customer will be split

into training sets and test sets. The training sets of each customer are finally batched together as the training pool. Same procedure is taken to form the test pool.

B. Stage 2: Pooling-Based Load Forecasting Using Deep Recurrent Neural Network

This stage of the proposed method consists of training and testing of pooling-based load forecasting: 1) In the training part, the deep recurrent neural network is trained by load profile batches randomly fetched from the load profile pool, so that deep-RNN are not only learning individual load patterns but also common sharing load features and uncertainties. 2) In the testing part, test load profiles are fed forward into the well-trained deep-RNN network.

Assuming the cleaned load profile database is Ψ_1, and the N testing households are listed in set $C = \{c_1, c_2, \ldots, c_N\}$. The deep RNN configuration parameters are specified with L and H, which represent the network depth (number of layers) and amount of hidden units. With these parameters, the training and testing process can be conducted in following steps.

1) Initiation of Deep Recurrent Neural Network: At the beginning, the deep recurrent neural network is built with network configuration parameters, i.e., the network depth L, amount of hidden units H, batch size B, input sequence size I, and output sequence size O.

2) Network Training Iterations: After network initiation, the program is then running training iteration epochs until the network is well-trained with converged network prediction loss in the form of reduced mean squared error (RMSE).

$$Loss\left(y_{predict}, y_{target}\right) = {}^{2}\sqrt{\frac{1}{B}\cdot\frac{1}{O}\cdot\sum_{i=1}^{B}\sum_{j=1}^{O}\left(y_{predict}-y_{target}\right)^{2}} \tag{7}$$

In each of its training epochs, the training batch is firstly fetched from the load profile pool, then fed into the deep recurrent neural network. Each training batch is two matrices with fixed size, i.e., input matrix with size $B \times I$ and output matrix with size $B \times O$.

The time-cost and iteration epochs of training process highly depend on feed-in data sequence size I, the choice of optimizer, network size (L, H) and training batch size B. In order to strike a well balance between training efficiency and efficacy, the training batch size B is variant during training: 1) at early epochs, B is set as a small number in order to approach the optimum point rapidly. 2) Then B is gradually increasing towards better training performance but sacrifices time cost.

3) Testing Iteration and Performance Benchmarking: The well-trained deep recurrent neural network is then tested on individual households by performing as a feed-forward prediction neural network. In the testing process, load forecasting is conducted on testing households one by one, to identify whether the proposed methods can achieve a performance improvement of load forecasting individually. In each of the iterations, a performance comparison is made with other load forecasting methods, including ARIMA, SVR, RNN and deep-RNN, which only trained with load profile data from the testing household.

IV. Implementation

This section introduces the implementation of the proposed methodology, including hardware, software platforms, and the program design.

A. Data Description

The data used in this paper are from the Smart Metering Electricity Customer Behaviour Trials (CBTs) initiated by Commission for Energy Regulation (CER) in Ireland. The trials took place during 1st July 2009 and 31st December 2010 with over 5000 Irish residential consumers and small and medium enterprises (SMEs) participating. The full anonymized data sets are publicly available online and comprise three parts: 1) half-hourly sampled electricity consumption (kWh) from each participant; 2) questionnaires and corresponding answers from surveys; 3) customer type, tariff and stimulus description, which specifies customer types, allocation of tariff scheme and Demand Side Management (DSM) stimuli [37].

In this trial, there were 929 1-E-E type consumers, meaning that they are all residential (1) customers with the controlled stimulus (E) and controlled tariff (E). To put it into perspective, these consumers were billed on existing flat rate without any DSM stimuli, which are most representative since the majorities of consumers outside the trial are of the type. In this paper, 920 1-E-E consumers were randomly selected as the testing customers. With group size 10, 920 consumers were split into 92 groups randomly.

Data with missing intervals are encountered and hence are not continuous. Different households may have different missing intervals and need to be pre-processed individually. Hardware and Software platforms.

B. Hardware and Software Platforms

The program is implemented on a high-performance Dell workstation equipped with Ubuntu 14.04 operating system and a computable GPU unit. The deep learning code is programmed based on an open-sourced deep learning framework named as Tensorflow [38], which is developed by one of the leading industry in the deep learning community, Google. Superior features of it include: 1) it is designed for the most popular programming language in data science, i.e., Python; 2) it supports GPU-based high-performance parallel computing towards big data tasks; 3) it employs symbolic programming mechanism and enables computing graph optimization feature, which is the most cutting-edge technique in deep learning community.

C. Program Implementation

The deep learning program is designed with multiple stages: 1) data pre-processing and cleaning; 2) data pooling; 3) data sampling and 4) network training and 5) benchmark

Program 1: Deep Learning Program for STLF

1: Load dataset Ψ_0 of household demand from smart meters.
2: Clean and pre-process demand data in dataset Ψ_1.
3: Generate tuple set $< L, H, C >$ of testing parameters: deep-RNN layer number $l \subseteq L$, deep-RNN hidden unit number $h \subseteq H$, and testing households set C.
4: **For** parameters $< l, h, c >$ in tuple set $< L, H, C >$:
5: According to household set C, get generate load profile pool $\Psi \subseteq \Psi_1$.
6: Divide Ψ into training set Ψ_{tr} and test set Ψ_{ts}.
7: Build deep-RNN $\aleph$ with network size (l, h) on Tensorflow.
8: **Repeat**
9: **At** k^{th} epochs **Do**:
10: Train deep-RNN $\aleph$ with randomly fetched data batch $\Phi \subseteq \Psi_{tr}$
11: Evaluate performance by mean squared error Λ_k on cross-validation samples.
12: Update a performance queue:

$$\Omega = [\Lambda_{k-v}, \Lambda_{k-v+1}, \ldots, \Lambda_{k-1}]$$

By pop out Λ_{k-v} from Ω, then push in Λ_k
13: **End**
14: **Until** $var(\Omega) \leq \varepsilon$, where ε is a convergence threshold.
15: **End**
16: **For** household c in set C:
17: Fetch test samples φ_c of household c from dataset Ψ_{ts}
18: Evaluate performance of $\aleph$ on test samples φ_c, with multiple performance benchmarks Θ
19: Compare load forecasting performance with other methods on household c.
20: **End**
21: **Terminate**

evaluation. The program design is demonstrated with pseudo code in Program 1.

D. Experiment Setup

This part presents the details for setting up the experiments, including data pre-process, algorithm configuration.

Regarding the data pre-process, raw data from Irish dataset is manipulated into input data sets through three steps: 1) split all customers into sub-groups; 2) construct load profile pool for each customer group; 3) split each pool into training, validation and test sets. The test set consist of data points during the last 30 days of available dataset (720 hours, 1440 data points), validation set is randomly selected from the rest of the data.

In order to reach optimal performance of each algorithm (SVR, ARIMA, RNN, DRNN, Pooling-based DRNN), we prepared multiple algorithm settings for each algorithm. However, not all results are reported in the result section, the comparison is made with the optimal settings of each algorithm.

In summary, all the experiment settings and parameters are presented as follows:

customer group size $\in \{10\}$
customer group amount $\in \{92\}$
test set size $\in \{1440\}$ *points of data*
validation set size $\in \{2880\}$ *points of data*
RNN network layer number $\in \{1\}$
DRNN network layer number $\in \{2, 3, 4, 5\}$
PDRNN network layer number $\in \{2, 3, 4, 5\}$
training batch size $\in \{96,\ 240,\ 480\}$ *points of data*
validation batch size $\in \{240,\ 480,\ 960\}$ *points of data*
hidden neuron number $\in \{5, 10, 20, 30, 50, 100\}$
input sequence length $\in \{48,\ 96, 336\}$
training method $\in \{AdamOptimizer\}$
neuron cell unit $\in \{LSTM\}$
learning rate $\in \{0.001,\ 0.002,\ 0.005,\ 0.01\}$
straining stop strategy $\in \{early\ stopping\}$
loss function $\in \{RMSE\}$.

V. Result and Discussion

In this section, the proposed method is validated on realistic smart metering load data from Irish load profile database [37]. The data selection and pre-processing are exploited in the data description section. To assess the performance of proposed method in conducting STLF for residential households, three widely used metrics are employed, including root mean squared error (RMSE), normalised root mean squared error, and mean absolute error [8].

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N} (\widehat{y_t} - y_t)^2}{N}} \tag{8}$$

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}} \tag{9}$$

$$MAE = \frac{\sum_{t=1}^{N} |\widehat{y_t} - y_t|}{N} \tag{10}$$

where $\widehat{y_t}$ is the forecasted value, y_t is the actual value, y_{max} and y_{min} is the maximum and minimum value among the test set. N refers to the test set size.

This assessment consists of three parts: 1) the performance of proposed method are compared to 3 methods and typical deep-RNN method to validate the efficacy; 2) the effect of network depth increase are illustrated to reveal the performance impact from 'shallow' to 'deep' architectures, to indicate the potential of deep learning for load forecasting and reveal the challenge of overfitting; and 3) the effect of pooling strategy are revealed by comparing proposed PDRNN typical with deep-RNN algorithm without pooling strategy, specifically to indicate the effect of pooling strategy to defer the overfitting issue.

A. Benchmarking of STLF Methods in Households

To validate the efficiency of the proposed PDRNN, three load forecasting methods, including autoregressive integrated moving average (ARIMA), support vector machine (SVR), and a 3-layer deep-RNN method are taken as a comparison and assessed under preceding mentioned benchmarks (RMSE, NRMSE, and MAE). The performance comparison across all testing residential households (920 households) is presented in Fig. 7 to Fig. 9 in form of heat map.

It is notable that the other 4 methods (RNN, SVR, DRNN, PDRNN) receive better average performance compared to ARIMA in the experiments. Therefore, we presents

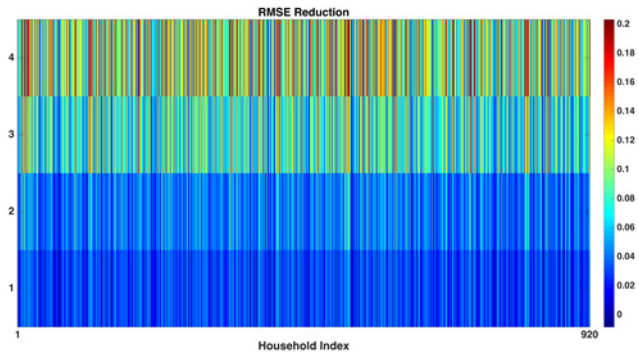


Fig. 7. RMSE reduction of 4 methods compared to ARIMA: 1) RNN, 2) SVR, 3) DRNN, 4) PDRNN.

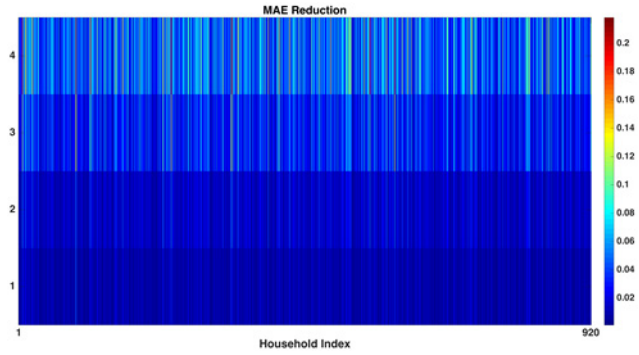


Fig. 8. MAE reduction of 4 methods compared to ARIMA: 1) RNN, 2) SVR, 3) DRNN, 4) PDRNN.

the performance improvement of 4 methods with respect to ARIMA method in the heat map. In the heat map, y axis refers to 4 methods (method 1: RNN, method 2: SVR, method 3: DRNN, method 4: PDRNN). x axis refers to 920 testing households. Lighter colour in the figure refers to better performance.

The results in Fig. 7 to Fig. 9 indicate that:

i) In terms of Average performance of three benchmarks, RNN and SVR achieve even performance, however, SVR performs slightly better than RNN in terms of RMSE and NRMSE. DRNN can receive a considerable improvement from RNN and SVR in all three benchmarks. The proposed PDRNN outperforms the other three methods, and can observe a clear reduction on all benchmarks.

ii) Regarding the results of different customers, the improvements of three benchmarks are not with same pattern. The improvements of RMSE among differing customer are largely diverse. While some customers receive 0.2 RMSE reductions, the other customers may receive only half of it. Unlike result of RMSE, the reduction of NRMSE and MAE are more consistent.

Furthermore, Fig. 10 demonstrates the real load and forecasted load by different methods on a random day 20 Jan. 2010, household 1059. The proposed method can deliver substantially improved performance at spikes and troughs. As shown in the figure, the morning peak during 8:00 a.m. and 10:00 a.m. is accurately captured by the proposed method. In addition, ARIMA, SVR, and 3-layer deep-RNN followed the inertia and predict a peak between

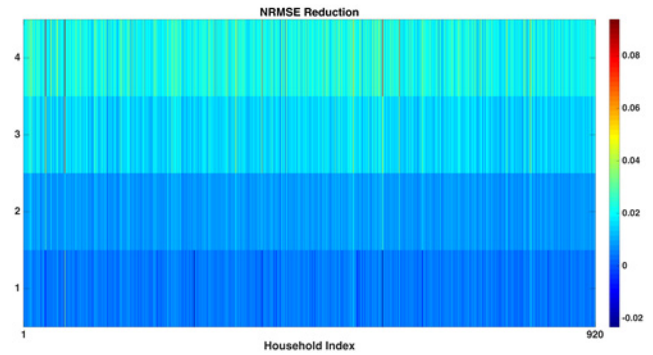


Fig. 9. NRMSE reduction of 4 methods compared to ARIMA: 1) RNN, 2) SVR, 3) DRNN, 4) PDRNN.

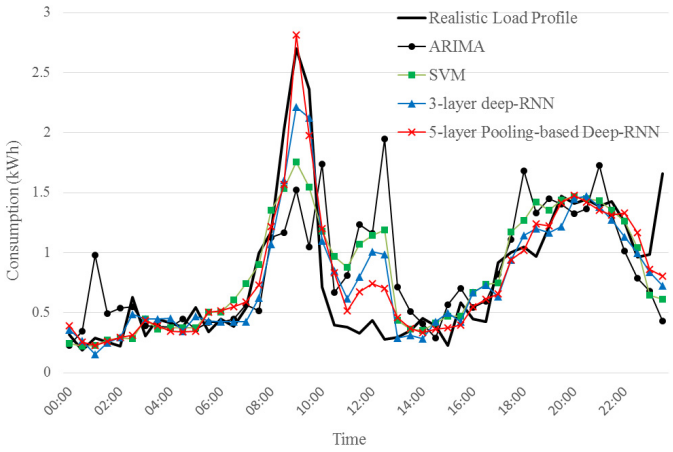


Fig. 10. The computational graph and unfolded topological graph.

10:00 a.m. and noon while the proposed method successfully avoids overestimating.

B. Effect From 'Shallow' to 'Deep'

A sensitivity analysis is conducted to investigate the effect of network depth on load forecasting performance, in terms of neural network based load forecasting methods. To make a fair assessment, recurrent neural networks with differing depth are all: 1) enhanced with LSTM units, 2) subjected to same input size, output size, network configuration parameters, and 3) implemented on Tensorflow with Python. The results are presented in Fig. 11.

In Fig. 11, deep RNN witnesses the best performance with 2 to 3 layers, with around 0.485 in *RMSE*, 0.27 in *R*, and 0.1 in *NRMSE*. Further increase in network depth will lead to severe overfitting issue. With 5 layers, deep-RNN gives even worse result than 1-layer RNN.

In general, the sensitivity analysis on network depth indicates that increasing network depth into 'deep' can only enhance the accuracy up to a limit number of layers, which reflects the occurrence of overfitting, due to the lack of data diversity and network parameter redundancy [39].

C. Effect of Proposed Pooling Strategy

The proposed pooling strategy attempts to tackle the occurrence of overfitting. The performance is investigated by

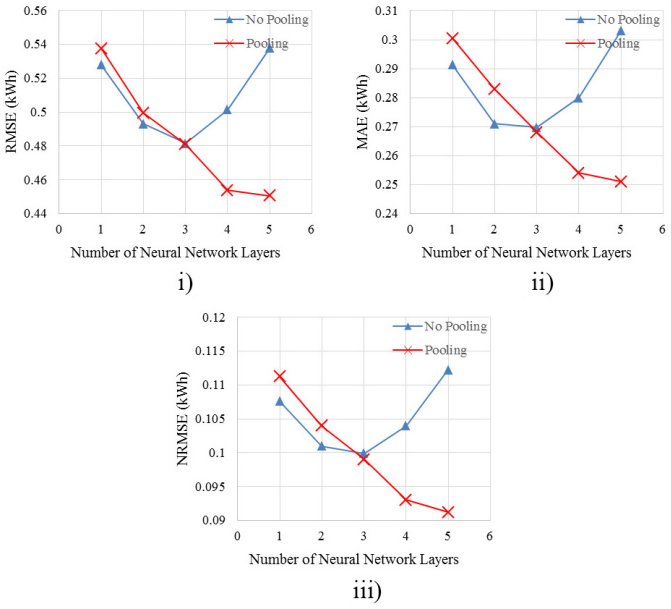


Fig. 11. Household load forecasting benchmarks from shallow to deep: i) root mean squared error (RMSE), ii) mean absolute error (MAE), iii) normalised root mean squared error (NRMSE).

TABLE I
PERFORMANCE COMPARISON

Network Architecture	*RMSE (kWh)*	*NRMSE (kWh)*	*MAE (kWh)*
ARIMA	0.5593	0.1132	0.2998
RNN	0.5280	0.1076	0.2913
SVR	0.5180	0.1048	0.2855
DRNN	0.4815	0.0974	0.2698
PDRNN	0.4505	0.0912	0.2510
Improvement from DRNN to PDRNN	6.45%		6.96%
Improvement from ARIMA to PDRNN	19.46%		16.28%

comparing the load forecasting performance between deep-RNN methods with and without pooling at different depths. The corresponding results are demonstrated in Fig. 11.

In Fig 11, the proposed PDRNN (red line marked with cross) are compared with classical deep RNN method (blue line marked with triangle). In terms of RMSE, MAE, NRMSE classical deep RNN's performance stops improve after 3 layers due to overfitting while the proposed method keeps improving as the number of layers increases till as deep as we tested.

Table I compares the performance of the proposed PDRNN in terms of RMSE, NRMSE, and MAE with four other techniques, i.e., classical DRNN, SVR, shallow RNN and ARIMA. All the presented metrics in the table take the averaged values across all the tested households. As illustrated, DRNN outperforms SVR, shallow RNN and ARIMA in all metrics used. With the introduction of the proposed pooling strategy, the new PDRNN with the same network settings (5 layers, with 30 hidden units in each layer), could further improve the performance. Specifically, compared with classical DRNN, the proposed PDRNN brings 6.45 % reduction in RMSE and NRMSE, 6.96 % reduction of MAE. Compared with ARIMA, the reduction in RMSE and MAE brought by PDRNN is even more significant, reaching 19.46% and 16.28% respectively.

VI. CONCLUSION

This paper for the first time explores the potential of employing the state-of-art deep learning technique for household STLF under high uncertainty and volatility. A novel PDRNN is proposed to successfully address the overfitting challenges brought by the naive deep network. This paper proposes method enables learning of spatial information shared between interconnected customers and hence allowing more learning layers before the occurrence of overfitting.

The result indicates the proposed method can deliver significant improvement for household load forecasting. Compared with state-of-the-art, the proposed method outperforms ARIMA by 19.5%, SVR by 13.1% and classical deep RNN by 6.5% in terms of RMSE and similar performance under other metrics.

Although quantitative comparison has been conducted, we would like to emphasize that we do not draw an arbitrary conclusion of the superiority of deep learning model. The key findings are the overfitting problem identified in direct applying deep learning models and the novel pooling methodology developed to overcome the limitation. The paper aims to report the preliminary attempt and provide learnings for wider researchers who aim to tap into this state-of-the-art technique. Future work includes:

i) To exploit the overfitting point by further extending the network size.

ii) To exploit optimal pooling strategy by pooling customers with differing features, such as similar geographic locations, similar social status.

iii) To further exploit the potential of proposed method by considering more external factors, for instance, weather information.

REFERENCES

[1] *Creating the Right Environment for Demandside Response*, Ofgem, London, U.K., Apr. 2013.
[2] *Demand Side Response Policy Paper*, Eur. Netw. Transm. Syst. Oper. Electricity, Brussels, Belgium, Sep. 2014.
[3] A. Garulli, S. Paoletti, and A. Vicino, "Models and techniques for electric load forecasting in the presence of demand response," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 3, pp. 1087–1097, May 2015.
[4] P. Du and N. Lu, "Appliance commitment for household load scheduling," *IEEE Trans. Smart Grid*, vol. 2, no. 2, pp. 411–419, Jun. 2011.
[5] X. Liu, L. Ivanescu, R. Kang, and M. Maier, "Real-time household load priority scheduling algorithm based on prediction of renewable source availability," *IEEE Trans. Consum. Electron.*, vol. 58, no. 2, pp. 318–326, May 2012.
[6] B. Stephen, X. Tang, P. R. Harvey, S. Galloway, and K. I. Jennett, "Incorporating practice theory in sub-profile models for short term aggregated residential load forecasting," *IEEE Trans. Smart Grid*, to be published.

[7] M. Chaouch, "Clustering-based improvement of nonparametric functional time series forecasting: Application to intra-day household-level load curves," *IEEE Trans. Smart Grid*, vol. 5, no. 1, pp. 411–419, Jan. 2014.

[8] C. J. Willmott *et al.*, "Statistics for the evaluation and comparison of models," *J. Geophys. Res*., vol. 90, pp. 8995-9005, Sep. 1985.

[9] A. Marinescu, C. Harris, I. Dusparic, S. Clarke, and V. Cahill, "Residential electrical demand forecasting in very small scale: An evaluation of forecasting methods," in *Proc. 2nd Int. Workshop Softw. Eng. Challenges Smart Grid (SE4SG)*, San Francisco, CA, USA, 2013, pp. 25–32.

[10] S. Humeau, T. K. Wijaya, M. Vasirani, and K. Aberer, "Electricity load forecasting for residential customers: Exploiting aggregation and correlation between households," in *Proc. Sustain. Internet ICT Sustain. (SustainIT)*, Palermo, Italy, 2013, pp. 1–6.

[11] A. Veit, C. Goebel, R. Tidke, C. Doblander, and H.-A. Jacobsen, "Household electricity demand forecasting: Benchmarking state-of-the-art methods," in *Proc. 5th Int. Conf. Future Energy Syst.*, Cambridge, U.K., 2014, pp. 233–234.

[12] S. Haben, J. Ward, D. V. Greetham, C. Singleton, and P. Grindrod, "A new error measure for forecasts of household-level, high resolution electrical energy consumption," *Int. J. Forecast.*, vol. 30, no. 2, pp. 246–256, 2014.

[13] Y.-H. Hsiao, "Household electricity demand forecast based on context information and user daily schedule analysis from meter data," *IEEE Trans. Ind. Informat.*, vol. 11, no. 1, pp. 33–43, Feb. 2015.

[14] Y. Wang, Q. Xia, and C. Kang, "Secondary forecasting based on deviation analysis for short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 26, no. 2, pp. 500–507, May 2011.

[15] T. Hong and D. A. Dickey. (Aug. 1, 2016). *Electric Load Forecasting: Fundamentals and Best Practices*. [Online]. Available: https://www.otexts.org/elf

[16] F. L. Quilumba, W.-J. Lee, H. Huang, D. Y. Wang, and R. L. Szabados, "Using smart meter data to improve the accuracy of intraday load forecasting considering customer behavior similarities," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 911–918, Mar. 2015.

[17] T. Hong, P. Wang, A. Pahwa, M. Gui, and S. M. Hsiang, "Cost of temperature history data uncertainties in short term electric load forecasting," in *Proc. IEEE 11th Int. Conf. Probabilistic Methods Appl. Power Syst. (PMAPS)*, Singapore, 2010, pp. 212–217.

[18] T. Hong, P. Wang, and L. White, "Weather station selection for electric load forecasting," *Int. J. Forecast.*, vol. 31, no. 2, pp. 286–295, Apr./Jun. 2015.

[19] P. Wang, B. Liu, and T. Hong, "Electric load forecasting with recency effect: A big data approach," *Int. J. Forecast.*, vol. 32, no. 3, pp. 585–597, Jul./Sep. 2016.

[20] X. Sun *et al.*, "An efficient approach to short-term load forecasting at the distribution level," *IEEE Trans. Power Syst.*, vol. 31, no. 4, pp. 2526–2537, Jul. 2016.

[21] R. Li, C. Gu, F. Li, G. Shaddick, and M. Dale, "Development of low voltage network templates—Part II: Peak load estimation by clusterwise regression," *IEEE Trans. Power Syst.*, vol. 30, no. 6, pp. 3045–3052, Nov. 2015.

[22] A. Espasa and I. Mayo-Burgos, "Forecasting aggregates and disaggregates with common features," *Int. J. Forecast.*, vol. 29, no. 4, pp. 718–732, Oct./Dec. 2013.

[23] J. Nowotarski, B. Liu, R. Weron, and T. Hong, "Improving short term load forecast accuracy via combining sister forecasts," *Energy*, vol. 98, pp. 40–49, Mar. 2016.

[24] 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.

[25] R. Al-Otaibi, N. Jin, T. Wilcox, and P. Flach, "Feature construction and calibration for clustering daily load curves from smart-meter data," *IEEE Trans. Ind. Informat.*, vol. 12, no. 2, pp. 645–654, Apr. 2016.

[26] D. Shi, R. Li, R. Shi, and F. Li, "Analysis of the relationship between load profile and weather condition," in *Proc. IEEE PES Gen. Meeting Conf. Expo.*, National Harbor, MD, USA, 2014, pp. 1–5.

[27] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[28] A. Lusci, G. Pollastri, and P. Baldi, "Deep architectures and deep learning in chemoinformatics: The prediction of aqueous solubility for drug-like molecules," *J. Chem. Inf. Model.*, vol. 53, no. 7, pp. 1563–1575, 2013.

[29] C.-Y. Zhang, C. L. P. Chen, M. Gan, and L. Chen, "Predictive deep Boltzmann machine for multiperiod wind speed forecasting," *IEEE Trans. Sustain. Energy*, vol. 6, no. 4, pp. 1416–1425, Oct. 2015.

[30] E. Busseti, I. Osband, and S. Wong, "Deep learning for time series modeling," Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep. CS 229, 2012.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.iro.umontreal.ca/∼bengioy/dlbook

[32] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[33] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[34] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[35] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Vancouver, BC, Canada, 2013, pp. 6645–6649.

[36] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.

[37] "Electricity smart metering customer behaviour trials (CBT) findings report," Commission Energy Regulation, Dublin, Ireland, Tech. Rep. CER11080b, 2011.

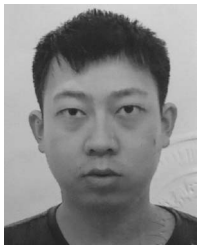
[38] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[39] D. M. Hawkins, "The problem of overfitting," *J. Chem. Inf. Comput. Sci.*, vol. 44, no. 1, pp. 1–12, 2004.

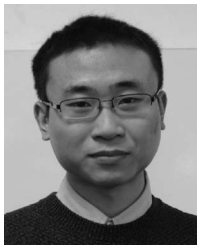
[40] T. Hong, P. Pinson, and S. Fan, "Global energy forecasting competition 2012," *Int. J. Forecast.*, vol. 30, no. 2, pp. 357–363, Apr./Jun. 2014.

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

[42] 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. [Online]. Available: http://dx.doi.org/10.1016/j.ijforecast.2015.11.011



Heng Shi was born in Hunan, China. He received the B.Eng. degree in computer science and technology from Tsinghua University, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree with the University of Bath. His major research interest is the big data analysis and deep learning applications in power system, especially in the LV distribution networks under high uncertainties.



Minghao Xu received the B.Eng. degrees in electrical and electronic engineering from the University of Bath, U.K., and electrical power engineering from North China Electric Power University, Baoding, China, in 2014. He is currently pursuing the Ph.D. degree with the University of Bath. His research interests include big data, machine learning, and deep learning applications in power systems.

Ran Li received the B.Eng. degrees in electrical power engineering from the University of Bath, U.K., and North China Electric Power University, Beijing, China, in 2011, and the Ph.D. degree from the University of Bath in 2014, where he is a Lecturer/Assistant Professor. His major interest is in the area of big data in power system, load profiling and forecasting, and power market and economics.