Short-term Load Forecasting with Distributed Long Short-Term Memory

Yi Dong
Department of Computer Science
University of Liverpool
Liverpool, the UK
yi.dong@liverpool.ac.uk

Yang Chen
Department of R & D
Suzhou SeeEx Technology Co., Ltd
Jiangsu, China
yang.chen@seeextech.com

Xingyu Zhao, Xiaowei Huang
Department of Computer Science
University of Liverpool
Liverpool, the UK
{xingyu.zhao, xiaowei.huang}@liverpool.ac.uk

Abstract—With the employment of smart meters, massive data on consumer behaviour can be collected by retailers. From the collected data, the retailers may obtain the household profile information and implement demand response. While retailers prefer to acquire a model as accurate as possible among different customers, there are two major challenges. First, different retailers in the retail market do not share their consumer's electricity consumption data as these data are regarded as their assets, which has led to the problem of data island. Second, the electricity load data are highly heterogeneous since different retailers may serve various consumers. To this end, a fully distributed short-term load forecasting framework based on a consensus algorithm and Long Short-Term Memory (LSTM) is proposed, which may protect the customer's privacy and satisfy the accurate load forecasting requirement. Specifically, a fully distributed learning framework is exploited for distributed training, and a consensus technique is applied to meet confidential privacy. Case studies show that the proposed method has comparable performance with centralised methods regarding the accuracy, but the proposed method shows advantages in training speed and data privacy.

Index Terms—short-term load forecasting, long short term memory, distributed learning, consensus, multi-agent system

I. INTRODUCTION

Electricity load forecasting is an essential basis for not only industrial production but also social life. The analysis of these load data could help in revealing household profile information and enabling other uses [1]–[3]. Specifically, retailers could obtain the electricity consumption behaviour of consumers and provide social and behavioural incentive signals to optimise customers' electricity usage. However, customers' data are not able to be shared among different power supply companies. Therefore, an appropriate algorithm that can generate accurate short-term load forecasting (STLF) while protecting custom privacy is essential for power system operation.

Load forecasting has been an essential part of the field of power system research since the 1970s, starting with some of the earliest methods, such as linear regression (LR) [4], stochastic time series (STS) [5] and general

exponential smoothing (GES) [6]. The mean accuracy of these methods is about 90%∼96%. With the introduction of machine learning and artificial intelligence methods, scholars started applying artificial intelligence algorithms to STLF and achieved great success, which increased the accuracy to more than 97% [2], [7], [8].

Due to the vigorous development of hardware equipment and the superb performance of deep learning in terms of the characteristics of nonlinear functions, deep learning algorithms have become particularly popular in load prediction research in recent years. [9] proposed a novel model multi-scale convolutional neural network with time-cognition (TCMS-CNN), which combines sufficient and discriminative features to extract potential law in the dataset providing an excellent result. [10] proposed a deep learning framework based on a combination of a convolutional neural network (CNN) and long short-term memory (LSTM), which could provide a significant improvement in the accuracy of individual household load forecasting. [11] improved deep belief networks (DBN) with Gauss-Bernoulli restricted Boltzmann machine (GB-RBM) and gray relational analysis (GRA). The developed method has a better performance than DBN and other traditional methods. However, the dataset required by the aforementioned methods includes all customers' data together without considering individual customers' privacy.

With the rapid development of big data, customer data privacy has gradually received attention. Over the last few years, several governments also have committed to data privacy protection, e.g., the European Commission's General Data Protection Regulation [12] and the Consumer Privacy Bill of Rights in the US [13]. As a result, *decentralised* learning methods, such as federal learning, have been proposed to solve these problems. [14] proposed a federated learning method for load forecasting with smart meter data that is capable of training a machine learning model in a distributed manner without requiring the participant to share their local data. [15] proposed a fully distributed STLF method based on Distributed Deep Belief Networks (DDBN), which can solve the STLF model by local computing agents (CA) and update the model parameters by com-

municating with connected neighbours. However, these distributed learning algorithms still require a data centre to communicate with individual nodes to update global parameters. In the case of huge data volumes, the central-based distributed method could cause traffic jams and scalability issues.

To this end, a novel fully distributed LSTM-based STLF method is proposed in this paper. It can be trained with a local dataset and converge to global optima by communicating with its connected neighbours, which does not need to transfer any customers' data. Different from the federated learning framework, the proposed consensus-based framework does not have a single computing centre to deal with an integrated global model among all local agents, and therefore the proposed algorithm will not meet computing burdens. Apart from data privacy and communication congestion, the distributed training framework can also avoid the local over-fitting problem since its parameters will be amended based on its neighbouring information.

The major contributions of this paper include:
1) A novel distributed LSTM algorithm is proposed for STLF, which can train the model at local and only communicate with its connected neighbours, preserving data privacy.
2) Due to the parallel computational framework, the proposed distributed method can significantly reduce the training time.
3) The over-fitting problem of the local model can be mitigated by mutual correction during the communication and training process.

The rest of the paper is organised as follows. Mathematical preliminaries applied in this paper are summarised in Sec. II. The proposed distributed LSTM method is described in Sec. III. Simulation results with discussion are presented in Sec. IV. Finally, we conclude the work in Sec. V.

II. APPLIED METHODOLOGIES

In this section, we recall methods regarding graph theory, consensus control and LSTM. Let $\mathbb{R}^{n \times m}$ be the set of $n \times m$ real matrices and the superscript T means the transpose of real matrices. I_N denotes the identity matrix of dimension N and $\mathbf{1}_N$ represents a column vector with all entries being 1. $\mathbb{R}^{++}$ denotes the positive real numbers. $\| \cdot \|_2$ represents the 2-norm of the argument.

A. Graph Theory

Following [16], an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be used to describe the communication topology among the local computing centres, where $\mathcal{V} = \{\nu_1, \cdots, \nu_N\}$ is the vertex set and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the edge set. The adjacency matrix $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$ of $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is an $N \times N$ matrix, such that $a_{ij} = 1$ if $(\nu_j, \nu_i) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. Define the degree matrix $D = \text{diag}\{\sum_{j=1}^{N} a_{1j}, \sum_{j=1}^{N} a_{2j}, \cdots, \sum_{j=1}^{N} a_{Nj}\}$. A graph is connected if and only if every
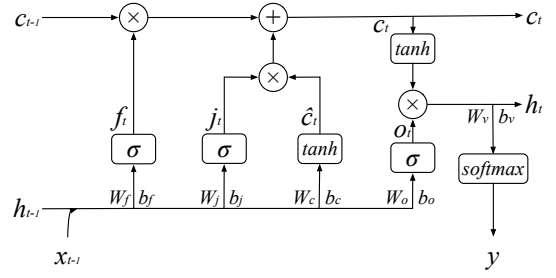

Fig. 1: Structure of LSTM.

pair of vertices can be connected by a path, namely, a sequence of edges. In this paper, we assume that the graph is connected and undirected. The Laplacian matrix related to $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})$ is defined as $\mathcal{L} = \mathcal{D} - \mathcal{A}$, i.e.,

$$\mathcal{L} = \begin{cases} l_{ij} = -a_{ij}, i \neq j \\ l_{ii} = \sum_{i \neq j} a_{ij}. \end{cases} \quad (1)$$

When $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a connected undirected graph, 0 is an eigenvalue of Laplacian $\mathcal{L}$ with the eigenvector $\mathbf{1}_N$ and all the other eigenvalues are positive. Then,

$$\mathcal{L}\mathbf{1}_N = 0_N, \quad \mathbf{1}_N^T \mathcal{L} = 0_N^T. \quad (2)$$

In this paper, we assume that the connection between computing centres are undirected since the communication line has no direction.

B. Long Short-Term Memory

The LSTM is a refined model of Recurrent Neural Network (RNN), and it has the advantage of avoiding exploding and vanishing gradient problem [17]. The basic structure of an LSTM network is shown in Fig. 1.

The propagate steps of LSTM can be separated into four sub-steps: *forget gate* step, *input gate* step, *update* step and *output* step. The *forget gate* step is made by a sigmoid layer which is to decide what information we are going to forget. The inputs of this step are historical output h_{t-1} and current state x_t, and it outputs a number within a range $(0, 1)$.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$

where W_f, b_f are the weights and bias of the LSTM forget gate layer, and σ denotes the sigmoid activation function.

The second step is composed of a sigmoid layer and a tanh layer. This sigmoid layer is to decide what new information we are going to memorise, and the tanh layer is to estimate the current cell state $\hat{C}_t$:

$$j_t = \sigma(W_j \cdot [h_{t-1}, x_t] + b_j) \quad (4)$$
$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (5)$$

The third step is to combine the outputs of second step and update the cell state:

$$C_t = f_t \times C_{t-1} + j_t \times \hat{C}_t \quad (6)$$

Finally, the LSTM yields the output controlled by the *output* step, which is based on the current cell state C_t, history output h_{t-1} and current input x_t. It uses a tanh activate function as a filter to push the cell state between

-1 and 1, so that the *output* step can only output the parts we choose.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{7}$$
$$h_t = o_t \times \tanh(C_t) \tag{8}$$

There is the final output y if end of the hidden layers:

$$y = softmax(W_v \cdot h_t + b_v) \tag{9}$$

III. DISTRIBUTED LONG SHORT-TERM MEMORY

In this section, we formulate a Distributed Long Short-Term Memory (DLSTM) method for STLF by consensus-based approaches. It consists of a group of N local LSTM models distributed over a connected graph, where each local computing centre has its own local dataset and cannot be revealed to other computing centres. The objective of the LSTM models is to minimise the empirical loss over the entire data set, which is formulated as

$$\min_{\Theta \in \mathbb{R}^n} \mathbb{E}(\mathcal{D}, \Theta) = \sum_{i=1}^{N} \mathbb{E}_i(\mathcal{D}_i, \Theta_i) \tag{10}$$

where $\mathcal{D}_i$ and $\Theta_i \in \mathbb{R}^n$ are the sub-dataset and weight parameters of the ith local LSTM model agent, respectively. n is the dimension of the weight matrix. It is noticed that the weight parameters Θ_i includes all the parameters in LSTM models, include *forget gate*, *input gate*, *update* and *output* steps:

$$\Theta_i = [W_f, b_f, W_j, b_j, W_C, b_C, W_o, b_o]^T \tag{11}$$

In the traditional LSTM algorithm, the training process is to calculate the gradient and update all the weights Θ:

$$\Theta^t = \Theta^{t-1} - \eta_i \nabla_\Theta \mathbb{E}(\mathcal{D}, \Theta^{t-1}) \tag{12}$$

For the DLSTM algorithm, the gradient descent method is applied to optimise the parameters of local LSTM models, and the distributed consensus algorithm is applied to optimise the parameters between connected LSTM models. All the communication and training processes only transfer the weights of different models, which does not require any customer data, and therefore the privacy can be protected. The procedure code of DLSTM is shown in Algorithm 1.

During each training step of the local computing agent, we add an communication update step for each local LSTM model. Thus, the gradient-based training of DLSTM can be formulated as the following sub-steps:

$$\Phi_i^t = \Theta_i^{t-1} - \eta_i \nabla_\Theta \mathbb{E}_i(\mathcal{D}_i, \Theta_i^{t-1}) \tag{13}$$
$$\Theta_i^t = \sum_{j \in \mathbb{N}} a_{ij} \Phi_i^t \tag{14}$$

where Φ_i^t is the intermediate variable and $\eta_i \in [0,1]$ is the learning rate of the local model. a_{ij} is the element of Laplace matrix $\mathcal{L}$ in (1). The updates of weights Θ_i include two procedures, which make up the learning before consensus (LBC) algorithm. In the first stage, the neural networks are trained independently with local sub-dataset through the gradient descent algorithm (13). This stage only uses local information, such as the weights of latest step Θ_i^{t-1} and the gradient of the empirical loss $\nabla_\Theta \mathbb{E}_i(\mathcal{D}_i, \Theta_i^{t-1})$. In the second stage, the weights of the neural networks are updated via

---

Algorithm 1 Distributed Long Short Term Memory.

1: **procedure** DLSTM(LSTM, Agents)
2:     $LSTM \leftarrow$ Initial weights Θ_i for local LSTM models
3:     $Sites \leftarrow$ Array of Network pipes to local CA_i
4:     **while** *Sites* contains unused data **do**
5:         //Start local training for each local site
6:         **for** each $CA_i \in$ *Sites* **do**
7:             $CA_i \leftarrow$ Forward Propagation
8:             $CA_i \leftarrow$ Gradient Descent Calculation
9:             $CA_i \leftarrow$ Back Propagation with (14)
10:         **end for**
11:         //Consensus the weight of local LSTM models
12:         **while** weights are different **do**
13:             **for** each $CA_i \in$ *Sites* **do**
14:                 Consensus weight with (14)
15:             **end for**
16:         **end while**
17:         $DLSTM \leftarrow$ Consensus weight
18:     **end while**
19: **end procedure**

---

consensus algorithm, where neighbouring information Φ_i^t are applied. With this learning process, all the agents are able to obtain the single and optimal neural network model when t tends to ∞ [18]. Similarly, the equations (14) and (13) can be swapped to yield the consensus before learning (CBL) algorithm as

$$\Phi_i^t = \sum_{j \in \mathbb{N}} a_{ij} \Theta_i^{t-1} \tag{15}$$
$$\Theta_i^t = \Phi_i^t - \eta_i \nabla_\Theta \mathbb{E}_i(\mathcal{D}_i, \Phi_i^t) \tag{16}$$

Note that the CBL algorithm is to consensus the gradients first, then update the local model. In contrast, the LBC algorithm is to update the local model first, then consensus the models. Theoretically, the gradients will converge to 0, and the model will become the optimal model with the training time goes infinity. Therefore, there is not much difference between CBL and LBC algorithms.

Based on the proposed distributed manner, each local LSTM model only needs to communicate with its neighbours. Thus, the STLF model can be trained locally and the multi-agent framework can reduce the computational and communication cost, which can be further extended to large networks [19]. Moreover, the distributed framework is more robust to single-point failures as long as the communication network remains connected [15].

IV. CASE STUDY

A. Experiment and Model Setup

The case study is based on the historical load data, daily average temperature data and holiday type data for the period 2016-2019 which are provided by the GEFCom 2017 competition [20] and ISO New England [21]. The system topology under consideration is the connection of four agents (data centres) as shown in Fig. 2, each of them has a quarter of dataset. In addition,

the iteration times of consensus between each information connection is set as 20, which makes the algorithm converge faster.
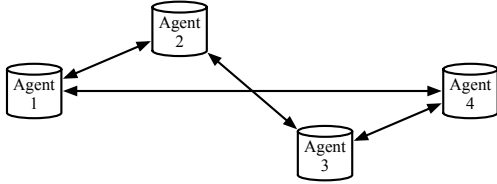

Fig. 2: Communication Topology.

For STLF, the main influencing factors are historical load, temperature, date type and other variables. These three types of data are selected as input variables for the forecasting model. The load has a characteristic of a 24-hour periodic change, so that the power load value at the same time before the forecasting day and two days before can be used as the input variable of the model. In addition, considering the effects of temperature on the electrical load, daily mean temperature after preprocessing can be used as input variables for the LSTM. The input and output information designed in this paper are summarised in Table I.

input	input variables
1-7	the electricity consumption of last week
8-14	the average temperature of last week
15-21	the day type of last week
22	the electricity consumption at 2 days ago
23	the average temperature at 2 days ago
24	the day type at 2 days ago
25	the actual load at 24 hours before
26	the average temperature at 24 hours before
27	the day type at 24 hours before
28	the temperature of the forecast day
29	the day type of the forecast day
output	the predicted load value at time t of the forecast day

TABLE I: Input and output information.

Further, all raw data are pre-normalised to improve the accuracy of the forecasting model:

$$x_i(k) = \frac{x_i(k) - x_i^{min}}{x_i^{max} - x_i^{min}} \qquad (17)$$

where x_i^{min} and x_i^{max} are the minimum and maximum value of the i-th original input data.

B. Evaluation Criteria and Error Analysis

The mean square error (MSE) and mean absolute percentage error (MAPE) are adopted as the final evaluating indicators, which are shown as follow:

$$\varepsilon_{MAPE} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\%, \qquad (18)$$

$$\varepsilon_{MSE} = \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i)^2}, \qquad (19)$$

where y_i is the actual load at period i; N is the total length of forecasting periods; and $\hat{y}_i$ is the forecast load at time i.

C. Results and Analysis

1) Case 1: This case verified the validity of the proposed DLSTM model. We use whole year customer electricity consumption data as the training dataset, and the test data is a dataset of a random week in next year. The results are shown in Fig. 3.
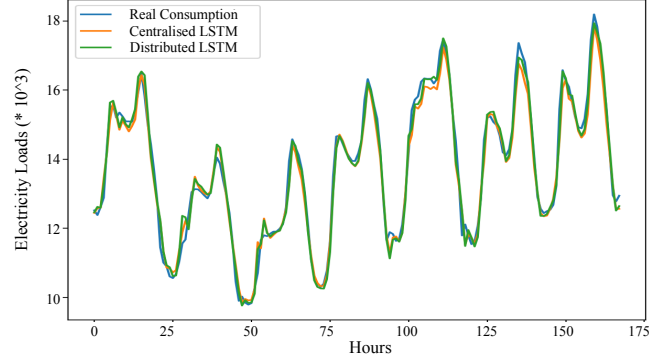

Fig. 3: Short-term Load Forecasting Results.

The green line is the forecasting results of the proposed DLSTM method, and it is evident that the proposed DLSTM has less prediction errors at most of the data points than centralised LSTM method.

2) Case 2: In this case, we investigate the training performance of the proposed DLSTM model. Fig. 4 depicts the validation errors during the training process among 4 distributed local DLSTM agents and centralised LSTM agent. Although each DLSTM agent only has a quarter of the dataset, it shows comparable training performance as the centralised LSTM method.
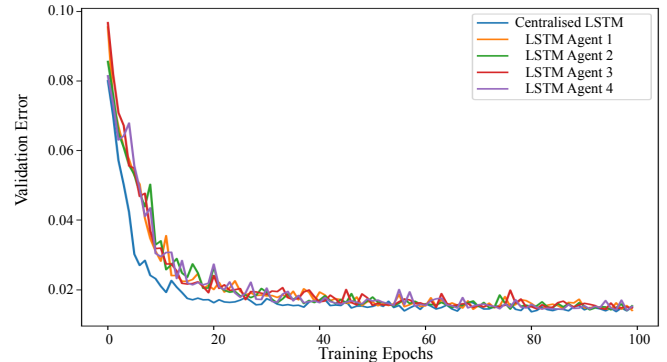

Fig. 4: Distributed Training Performance.

In Fig. 4, we may clearly see that the converge speed of local computing LSTM models are slower than centralised LSTM model, but they can achieve similar accuracy after sufficient training epochs. Apart from the accuracy, the training speed of the proposed DLSTM is only 412 seconds, which is much faster than centralised LSTM (583 second). This is because that the dataset size of local training agent is only a quarter of centralised LSTM agent. Therefore, the proposed DLSTM algorithm can reduce the training time with an increasing number of computing agents, and the aim of fast training can be

achieved by adding more computing agents if the load dataset continues to increase in the future.

3) Case 3: In this case, we summarise different STLF models to illustrate the advantage of the proposed model, and four typical state-of-the-art algorithms are chosen for comparison: Support Vector Regression (SVR), Artificial Neural Network (ANN), Deep Belief Network (DBN) and Distributed Artificial Neural Network (DANN). The compared models are simulated based on the same dataset in this paper. All the models are trained with the same training dataset, validation dataset and test dataset. The results of different models are summarised in Table II.

	MAPE	MAE	MSE
LSTM	0.0150	201.746	70021.160
DLSTM	0.0138	183.352	62095.324
SVR	0.0152	202.246	73495.060
ANN	0.0280	324.650	131273.362
DBN	0.0206	144.326	44299.052
DANN	0.0275	284.240	123363.703

TABLE II: Comparison of STLF Models in Case 3.

From the results provided in Table. II, the DLSTM algorithm meets the requirements of STLF and has better accuracy than other models. Comparing with the distinct machine learning algorithms, the LSTM and DLSTM reveals better performance and less prediction errors due to the time-sequence characteristic of STLF problems. For distributed algorithms, we can see that the DANN and DLSTM show the similar accuracy as ANN and LSTM, respectively. It can be seen from the Table. II and Fig. 4, the proposed DLSTM method demonstrates similar and even better accuracy than traditional single centralise LSTM algorithm after around 50 training epochs.

V. Conclusion and Future Work

This paper ascertains the effectiveness of using the distributed long short-term memory models in short-term load forecasting. The multiple and time-variable variations can be predicted by the proposed approach. The decentralised protocol makes it possible to separate data sets and pre-train models at an individual data centre, which simplifies the forecasting problem and leads to more privacy results. In the application examples, the proposed distributed long short-term memory model provided accurate forecast results and faster training speed than other machine learning models. The distributed model is simpler to train and tune than the centralised models, does not over-fit and reduces variance due to the consensus weight of many subsets. In future, we plan to apply trustworthy AI techniques, e.g. [22]–[24], to assure the reliability and security of our method, as well as its explainability.

References

[1] S. N. Fallah, M. Ganjkhani, S. Shamshirband, and K.-w. Chau, "Computational intelligence on short-term load forecasting: A methodological overview," *Energies*, vol. 12, no. 3, p. 393, 2019.

[2] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on power systems*, vol. 16, no. 1, pp. 44–55, 2001.

[3] J. W. Taylor and P. E. McSharry, "Short-term load forecasting methods: An evaluation based on european data," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 2213–2219, 2007.

[4] A. D. Papalexopoulos and T. C. Hesterberg, "A regression-based approach to short-term system load forecasting," *IEEE Transactions on power systems*, vol. 5, no. 4, pp. 1535–1547, 1990.

[5] M. T. Hagan and S. M. Behr, "The time series approach to short term load forecasting," *IEEE transactions on power systems*, vol. 2, no. 3, pp. 785–791, 1987.

[6] W. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems*, no. 2, pp. 900–911, 1971.

[7] C.-N. Lu, H.-T. Wu, and S. Vemuri, "Neural network based short term load forecasting," *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 336–342, 1993.

[8] S. Kiartzis, A. Bakirtzis, and V. Petridis, "Short-term load forecasting using neural networks," *Electric Power Systems Research*, vol. 33, no. 1, pp. 1–6, 1995.

[9] Z. Deng, B. Wang, Y. Xu, T. Xu, C. Liu, and Z. Zhu, "Multi-scale convolutional neural network with time-cognition for multi-step short-term load forecasting," *IEEE Access*, vol. 7, pp. 88 058–88 071, 2019.

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

[11] X. Kong, C. Li, F. Zheng, and C. Wang, "Improved deep belief network for short-term load forecasting considering demand-side management," *IEEE Transactions on Power Systems*, vol. 35, no. 2, pp. 1531–1538, 2019.

[12] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017.

[13] Sen. Markey, Edward J. [D-MA], "S.1214 - privacy bill of rights act," https://www.congress.gov/bill/116th-congress/senate-bill/1214#, April 2019.

[14] M. N. Fekri, K. Grolinger, and S. Mir, "Distributed load forecasting using smart meter data: Federated learning with recurrent neural networks," *International Journal of Electrical Power & Energy Systems*, vol. 137, p. 107669, 2022.

[15] Y. Dong, Z. Dong, T. Zhao, Z. Li, and Z. Ding, "Short term load forecasting with markovian switching distributed deep belief networks," *International Journal of Electrical Power & Energy Systems*, vol. 130, p. 106942, 2021.

[16] N. Biggs, N. L. Biggs, and E. N. Biggs, *Algebraic Graph Theory*. Cambridge university press, 1993, vol. 67.

[17] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5929–5955, 2020.

[18] Z. Li, B. Liu, Z. Dong, and Z. Ding, "Cooperative training over networks via consensus-based algorithms," in *2021 40th Chinese Control Conference (CCC)*. IEEE, 2021, pp. 5309–5314.

[19] Y. Dong, T. Zhao, and Z. Ding, "Demand-side management using a distributed initialisation-free optimisation in a smart grid," *IET Renewable Power Generation*, vol. 13, no. 9, pp. 1533–1543, 2019.

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

[21] K. B. Sahay and M. M. Tripathi, "Day ahead hourly load forecast of pjm electricity market and iso new england market by using artificial neural network," in *ISGT 2014*. IEEE, 2014, pp. 1–5.

[22] W. Huang, Y. Sun, X. Zhao, J. Sharp, W. Ruan, J. Meng, and X. Huang, "Coverage-Guided Testing for Recurrent Neural Networks," *IEEE Transactions on Reliability*, pp. 1–16, 2021.

[23] X. Zhao, A. Banks, J. Sharp, V. Robu, D. Flynn, M. Fisher, and X. Huang, "A Safety Framework for Critical Systems Utilising Deep Neural Networks," in *Computer Safety, Reliability, and Security*, ser. LNCS, vol. 12234. Cham: Springer, 2020, pp. 244–259.

[24] X. Zhao, W. Huang, X. Huang, V. Robu, and D. Flynn, "BayLIME: Bayesian local interpretable model-agnostic explanations," in *UAI'21*, vol. 161. PMLR, 2021, pp. 887–896.