Probabilistic Residential Load Forecasting with Sequence-to-sequence Adversarial Domain Adaptation Networks

Hanjiang Dong, *Graduate Student Member*, *IEEE*, Jizhong Zhu, *Fellow*, *IEEE*, Shenglin Li, Yuwang Miao, Chi Yung Chung, *Fellow*, *IEEE*, and Ziyu Chen

***Abstract*—Lately, the power demand of consumers is increasing in distribution networks, while renewable power generation keeps penetrating into the distribution networks. Insufficient data make it hard to accurately predict the new residential load or newly built apartments with volatile and changing time-series characteristics in terms of frequency and magnitude. Hence, this paper proposes a short-term probabilistic residential load forecasting scheme based on transfer learning and deep learning techniques. First, we formulate the short-term probabilistic residential load forecasting problem. Then, we propose a sequence-to-sequence (Seq2Seq) adversarial domain adaptation network and its joint training strategy to transfer generic features from the source domain (with massive consumption records of regular loads) to the target domain (with limited observations of new residential loads) and simultaneously minimize the domain difference and forecasting errors when solving the forecasting problem. For implementation, the dominant techniques or elements are used as the submodules of the Seq2Seq adversarial domain adaptation network, including the Seq2Seq recurrent neural networks (RNNs) composed of a long short-term memory (LSTM) encoder and an LSTM decoder, and quantile loss. Finally, this study conducts the case studies via multiple evaluation indices, comparative methods of classic machine learning and advanced deep learning, and various available data of the new residentical loads and other regular loads. The experimental results validate the effectiveness and stability of the proposed scheme.**

***Index Terms*—Domain adaptation, neural network, residential load forecasting, transfer learning, probabilistic forecasting.**

Manuscript received: November 1, 2023; revised: December 25, 2023; accepted: March 17, 2024. Date of CrossCheck: March 17, 2024. Date of online publication: May 10, 2024.

This work was supported by the National Natural Science Foundation of China (No. 52177087).

H. Dong, J. Zhu (corresponding author), S. Li, Y. Miao, and Z. Chen are with the School of Electric Power Engineering, South China University of Technology, Guangzhou, China, and H. Dong is also with the Department of Electrical Engineering, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: hanjiang.dong@foxmail.com; zhujz@scut.edu.cn; iamlshl@126.com; miaoyw2021@163.com; epchenzy@mail.scut.edu.cn).

C. Y. Chung is with the Department of Electrical Engineering, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: c.y.chung@polyu.edu.hk).

DOI: 10.35833/MPCE.2023.000841

I. Introduction

SHORT-TERM load forecasting of power demand is one of the main research areas in electrical engineering [1]. Load forecasts are crucial to the planning and operation of power systems, particularly in the context of developing electricity markets and promoting renewable energy. When integrating distributed renewables, generating accurate forecasts is one of the essential parts in the energy management of residential households [2] conducted by regional operators. For instance, the renewable power generation such as photovoltaic may provide part of the household power demand. The accurate load forecasts are important for the operators, especially in twofold contexts. First, the capacity of distributed solar power can occupy over 45% of the total capacity [3]. Then, residential prosumers generate around two-thirds of the growth in distributed solar power [4].

Numerous techniques have been applied to generate forecasts, which can be grouped into classic statistical techniques (e.g., stochastic time-series models [5]), machine learning (e.g., support vector regression [6]), and deep learning techniques (e.g., long short-term memory (LSTM) recurrent neural networks (RNNs)) [7]. Identifying an adequate model is the key to producing accurate forecasts and is the core link of the forecasting procedure [8]. Deep learning, which allows computational models consisting of multiple processing layers to learn representations of data with multiple levels of abstraction, has improved the state-of-the-art in financial indices, weather indices, energy demand, and other areas in addition to achieving breakthroughs in computer vision, natural language processing, and speech recognition [9]. The success of deep learning relies on the automation of discovering intricate neural networks (NNs) in high-dimensional features of each hidden layer in modularized, data-driven, and end-to-end manners instead of manually selecting a given dataset or given data sources [10].

Lately, the community has paid attention to residential load forecasting and probabilistic forecasting problems. On the one hand, the residential load is uncertain and random, depending on varying individual behavior patterns, regional layouts, and complicated factors such as weather conditions, exhibiting random time-series characteristics [11]-[14]. Reference [11] considers the practice theory of human behaviors to present a sampling model based on the Markov chain to predict the aggregated residential loads in a bottom-up style. References [12] and [14] use LSTM RNNs to process residential behaviors through the whole residential consumption and selected appliances, to generate short-term forecasts. Reference [13] disaggregates the total load of aggregated households into partial loads monitored by smart meters as well as others predicted via NNs. In summary, the data-driven methods, particularly LSTM RNNs, have exhibited superior generalization capability in generating residential load forecasts.

Furthermore, sequence-to-sequence (Seq2Seq) RNNs can describe the volatile temporal characteristics of residential consumption records in distribution networks [15]. Reference [16] explores Seq2Seq RNNs for modeling unpredictable supply-demand imbalances caused by the variable nature of distributed renewable power generation in commercial or office buildings. Reference [17] attempts to predict the peak residential demand supplied by distribution network operators, who handle the energy policy and deploy demand response programs, by designing bidirectional LSTM (B-LSTM) Seq2Seq RNNs. As deep learning methods make the energy industry to be more reliable and sustainable, [18] confirms the feasibility of Seq2Seq RNNs for producing multi-step load forecasting in single households. Hence, Seq2Seq RNNs serve as a promising solution to processing input sequences (e.g., influential factors) and output sequences (e.g., multi-step targets). The investigations and explorations of Seq2Seq RNNs in residential load forecasting are imperative.

On the other hand, the probabilistic load forecasting solutions that can describe the uncertainty of predictions are upsurging [19]-[21]. Specifically, [20] proposes a convolutional neural network (CNN) with squeeze-and-excitation modules to handle micrometeorological records and obtain day-ahead probabilistic load forecasts for residents. Reference [21] proposes a modified memristive LSTM RNN by characterizing variable-wise features and the temporal importance via a mixture of attention-based techniques to interpret the predictions produced by the model. In summary, the probabilistic load forecasting solutions are increasingly significant for households, particularly when integrating stochastic and intermittent renewables. However, there is a lack of the exploration of utilizing the adaptive Seq2Seq RNNs to get flexible probabilistic forecasts, illustrating the research gap in this topic.

Moreover, current studies seldom notice the limited implementation of the powerful Seq2Seq RNNs to the forecasting scheme for the new residential loads that generally lack enough data records. In other words, insufficient data make it difficult to generate accurate forecasts for the new residential load. To our knowledge, there remains no technical work focusing on new residential loads from real-life considerations, which motivates us to propose a probabilistic residential load forecasting scheme using limited data on the Seq2Seq RNN. The domain adaptation network is well-known in computer vision and other artificial intelligence fields, but its application to the residential load forecasting is not so well-known from expectation. Reference [22] presents an ensemble model, where the updated data can be utilized to adjust the weights by a sample domain adaptation method called Tradaboost. Reference [23] improves the adversarial domain adaptation method through an initial state fusion strategy that analyzes adversarial disequilibrium and an information entropy index that quantifies domain similarity. In this context, we recognize the realistic condition, where sufficient data of regular loads cannot be directly used through deep learning methods to generate forecasts for the new residential load due to distribution shift. In other words, the limited data of new residential loads are not enough for leveraging the power of deep learning methods, and deep learning methods should not be directly used due to the assumption that the training data are independent and identically distributed and the risk of overfitting.

Hence, this paper proposes a short-term probabilistic residential load forecasting scheme. The aim of this study is to solve the load forecasting problem for the new residential loads, where many other regular loads have been connected to power system for a relatively long time. The distribution of energy demand of the new residential loads, which is difficult to approximate with insufficient or limited data, can be different from that of other regular loads that own relatively sufficient data. The innovation of the proposed scheme is the Seq2Seq adversarial domain adaptation network that can align distributions of sufficient data of regular loads and the limited data of new residential loads, thus reasonably meeting the assumption that the training data are independent and identically distributed before leveraging the potential power of deep learning methods to improve the forecasts, which is necessary but usually ignored. Thus, the proposed scheme can process sufficient data of regular loads to maintain model performance, though the data of the new residential loads are limited.

The contributions of this paper are as follows.

1) A Seq2Seq adversarial domain adaptation network and its joint training strategy are proposed to align the distribution of sufficient data of regular loads and the limited data of the new residential loads.

2) An LSTM cell based teacher forces the Seq2Seq RNN with quantile loss as the submodules of the Seq2Seq adversarial domain adaptation network to capture temporal dependencies between regular loads and the new residential loads.

3) The proposed Seq2Seq adversarial domain adaptation network is comprehensively compared with other classic methods in different horizons, criteria, and scarcity degrees of energy consumption observations.

The remainder of this paper is organized as follows. Section II formulates the short-term probabilistic residential load forecasting problem. Section III proposes the short-term probabilistic residential load forecasting scheme. Section IV performs a comprehensive case study to validate the proposed scheme using multiple evaluation indices, comparative techniques, and limited degrees. Section V draws the conclusions.

II. Formulation of Short-term Probabilistic Residential Load Forecasting Problem

This section will clarify the short-term probabilistic residential load forecasting problem from the short-term deterministic load forecasting problem and the conventional multi-step (such as 24-point day-ahead) load forecasting problem.

A. Short-term Deterministic Load Forecasting Problem

Given the dataset $\mathcal{O}_t^{\text{clean}}$ at the time t, we often process the chronological observations by identifying adequate models and optimizing the model assignments. Denote the current time as t_c, the forecasting gap time as t_g, the forecasting lead time (i.e., horizon) as t_h, and the number of horizons as N^{hor}. The records at a certain time $\boldsymbol{o}_t^{\text{clean}}$ cover the elements that can be classified into the static covariate vector $\boldsymbol{o}_t^{\text{static}}$, the historical vector $\boldsymbol{o}_t^{\text{past}}$, and the future vector $\boldsymbol{o}_t^{\text{future}}$, where $t = 1, 2, \ldots, t_c, \ldots, t_c + t_h, \ldots, t_c + N^{\text{hor}} t_h$. The static covariate vector $\boldsymbol{o}_t^{\text{static}}$ covers temporal-static factors such as the load location and the identity number of residents. The historical vector $\boldsymbol{o}_t^{\text{past}} = \left[\boldsymbol{o}_{t-t_w:t}^{\text{obs}}, \boldsymbol{o}_{t-t_w:t}^{\text{known}}\right]$ involves the information available from the historical time, i.e., the observed factors $\boldsymbol{o}_{t-t_w:t}^{\text{obs}}$ such as target loads and other public factors $\boldsymbol{o}_{t-t_w:t}^{\text{known}}$, where t_w is denoted as the span of the last records for each horizon. The future vector $\boldsymbol{o}_t^{\text{future}} = \left[\boldsymbol{o}_{t:t+t_g+t_h}^{\text{target}}, \boldsymbol{o}_{t:t+t_g+t_h}^{\text{known}}\right]$ involves the information for the future moments, i.e., the target vector $\boldsymbol{o}_{t:t+t_g+t_h}^{\text{target}}$ and other pre-known factors $\boldsymbol{o}_{t:t+t_g+t_h}^{\text{known}}$ such as calendar rules. In regression, the target vector $\boldsymbol{o}_{t:t+t_g+t_h}^{\text{target}}$ is the dependent variable $\boldsymbol{y}_t$, and the input variable $\boldsymbol{x}_t$ involves historical, future, and static covariate vectors, i. e., $f\colon \boldsymbol{x}_t \mapsto \boldsymbol{y}_t, \boldsymbol{x}_t = \left[\boldsymbol{o}_{t-t_w:t}^{\text{obs}}, \boldsymbol{o}_{t-t_w:t+t_g+t_h}^{\text{known}}, \boldsymbol{o}_t^{\text{static}}\right]$, $\boldsymbol{y}_t = \boldsymbol{o}_{t:t+t_g+t_h}^{\text{target}}$, as expressed in (1).

$$\mathcal{F}_i := \left\{ f\colon \boldsymbol{x}_t \mapsto \boldsymbol{y}_t \middle| \boldsymbol{y}_t = f\left(\boldsymbol{x}_t; \boldsymbol{\theta}_i\right), \boldsymbol{\theta}_i \in \mathbb{R}^{N^{\text{par}}} \right\} \tag{1}$$

where $f\colon \boldsymbol{x}_t \mapsto \boldsymbol{y}_t$ reflects the mapping between the input $\boldsymbol{x}_t$ and the output $\boldsymbol{y}_t$; $\mathcal{F}_i$ is the assignment space of the model $i, i = 1, 2, \ldots, N_i$, and N_i is the number of candidate models; $\boldsymbol{\theta}_i$ is the parameter vector or matrix of the model i; and N^{par} is the number of parameters.

Given the dataset D of influential factors and target variables, the experts estimate the optimal parameter assignments $\boldsymbol{\theta}_i^*$ for the alternative models over varying optimal hyperparameters $\boldsymbol{\lambda}_i^*$ that are tuned by trials and errors with the training data $D^{\text{est}} = \left\{\boldsymbol{x}_t^{\text{est}}, \boldsymbol{y}_t^{\text{est}}\right\}$, as expressed in (2). Let f_i be the mapping described via model i. Experts then compare the performance of the alternative models to identify the adequate model to produce predictions $\hat{\boldsymbol{y}}_t^{\text{est}}$ over the testing inputs $\boldsymbol{x}_t^{\text{est}}$ (a.k.a., features), assuming that true testing targets $\boldsymbol{y}_t^{\text{est}}$ remain unknown before deploying it, as shown in (3).

$$\boldsymbol{\theta}_i^* = \underset{\boldsymbol{\theta}_i \in \Theta_i}{\arg\min}\, R^{\text{emp}}\left(\boldsymbol{\theta}_i\right) \triangleq \underset{\boldsymbol{\theta}_i \in \Theta_i}{\arg\min}\, \mathbb{E}_{D^{\text{est}} \overset{\text{i.i.d.}}{\sim} \mathcal{D}}\left(\mathcal{L}\left(f_i\left(\boldsymbol{\theta}_i\right), D^{\text{est}} \middle| \boldsymbol{\lambda}_i^*\right)\right) \tag{2}$$

$$f^* = \underset{f \in \left\{f_i^*\right\}_{i=1}^{N_i}}{\arg\min}\, \mathbb{E}_{D^{\text{est}} \overset{\text{i.i.d.}}{\sim} \mathcal{D}}\left(\mathcal{C}\left(f_i, D^{\text{est}} \middle| \boldsymbol{\theta}_i^*, \boldsymbol{\lambda}_i^*\right)\right) \tag{3}$$

where Θ_i is the set of $\boldsymbol{\theta}_i$; $R^{\text{emp}}\left(\boldsymbol{\theta}_i\right)$ is the empirical loss; $\mathbb{E}_{D^{\text{est}} \overset{\text{i.i.d.}}{\sim} \mathcal{D}}\left(\mathcal{L}(\cdot)\right)$ is the expectation of the loss function $\mathcal{L}(\cdot)$ on the independent and identically distributed training data D^{est} under an unknown distribution $\mathcal{D}$; and $\mathcal{C}(\cdot)$ is a cost function to evaluate the forecasting accuracy.

B. Multi-step Load Forecasting Problem

When the lead time is a time step, the model generates a single scalar (i.e., point) at each time, where the load forecasting problem is defined as a single-step load forecasting problem in (4). When the lead time covers more than one observed moment for the whole next day or week, the model outputs a specific profile composed of multiple points at a time in the multi-step load forecasting problem. There are three major solutions to the multi-step load forecasting problem, i.e., the day-ahead forecasting problem in this study: ① combining multiple single-step load forecasting models, each of which produces forecasts independently, as depicted in (5); ② using a multi-step load forecasting model to generate interval-ahead curves once and for all, as given in (6), which is adopted in this study by deep learning methods; and ③ replacing the inputs with the last predicted values to generate the forecasts iteratively, as given in (7).

$$\mathcal{F}_i := \left\{ f\colon \boldsymbol{x}_t \mapsto y_t \middle| y_t = f\left(\boldsymbol{x}_t; \boldsymbol{\theta}_i\right), \boldsymbol{\theta}_i \in \mathbb{R}^{N^{\text{par}}} \right\} \tag{4}$$

$$\mathcal{F}_{i,h} := \left\{ f\colon \boldsymbol{x}_{t,h} \mapsto y_{t,h} \middle| y_{t,h} = f_i\left(\boldsymbol{x}_{t,h}; \boldsymbol{\theta}_{i,h}\right) \right\} \tag{5}$$

$$\mathcal{F}_i := \left\{ f\colon \boldsymbol{x}_t \mapsto \boldsymbol{y}_t \middle| \boldsymbol{y}_t = f\left(\boldsymbol{x}_t; \boldsymbol{\theta}_i\right) \right\} \tag{6}$$

$$\mathcal{F}_i := \left\{ f\colon \boldsymbol{x}_{t,h} \mapsto y_{t,h} \middle| y_{t,h} = f\left(\boldsymbol{x}_{t,h}; \boldsymbol{\theta}_i\right) \right\} \tag{7}$$

where h is the index of horizon, $h = 1, 2, \ldots, N^{\text{hor}}$.

C. Probabilistic Load Forecasting Problem

The single-step load forecasting problem is defined as a deterministic forecasting problem when the model generates a deterministic value at each time, as given in (4). The deterministic forecasts are the most common form but cannot portray the uncertainty of the predictions. Thus, uncertain predictions in the forms of intervals, quantiles, and distribution densities are targeted in the probabilistic load forecasting problem. Each model depicts the probability distribution $F\left(Y_t \middle| \boldsymbol{X}_t\right)$ with the random variable Y_t of the target variable y_t at the time t, given the random variable $\boldsymbol{X}_t$ of the input $\boldsymbol{x}_t$, i.e., $Y_t \sim F\left(Y_t \middle| \boldsymbol{X}_t\right)$. The random variable Y_t correlates to the conditional probability distribution $P_t^{(\cdot)}\left(y \middle| \boldsymbol{x}_t\right) = F\left(y_t \le y \middle| \boldsymbol{x}_t\right)$ for the target variable $Y_t \sim P_t^y\left(y_t \middle| \boldsymbol{x}_t\right)$ or the forecasting error $Y_t \sim y_t + P_t^{\varepsilon}\left(\varepsilon_t \middle| \boldsymbol{x}_t\right)$ given the input $\boldsymbol{x}_t$.

Moreover, the conditional probability distribution $P_t^{(\cdot)}\left(y \middle| \boldsymbol{x}_t\right)$ can be represented by the probability distribution density P_t in (8), the quantile point $y_{\alpha_q,t}$ in (9), or the random prediction interval I_t in (10) [24].

$$\mathcal{F}_i := \left\{ f\colon \boldsymbol{x}_t \mapsto P_t \middle| P_t\left(y \middle| \boldsymbol{x}_t\right) = f\left(\boldsymbol{x}_t; \boldsymbol{\theta}_i\right) \right\} \tag{8}$$

$$\mathcal{F}_{i,q} := \left\{ f\colon \boldsymbol{x}_{\alpha_q,t} \mapsto y_{\alpha_q,t} \middle| y_{\alpha_q,t} = f\left(\boldsymbol{x}_{\alpha_q,t}; \boldsymbol{\theta}_{i,q}\right) \right\} \tag{9}$$

$$\mathcal{F}_i := \left\{ f\colon \boldsymbol{x}_t \mapsto I_t \middle| I_t\left(y \middle| \boldsymbol{x}_t\right) = f\left(\boldsymbol{x}_t; \boldsymbol{\theta}_i\right) \right\} \tag{10}$$

where α_q is the nominal quantile level, and $q = 1, 2, \ldots, Q$ is the order of quantile prediction; $\boldsymbol{\theta}_{i,q}$ is the parameter vector or matrix of model i for the quantile q; and $I_t\left(y \middle| \boldsymbol{x}_t\right) =$

$\left\{y_{\underline{\alpha}_q,t}, y_{\bar{\alpha}_q,t}\right\}$ is the prediction interval composed of lateral quantiles, and $\bar{\alpha}_q$ and $\underline{\alpha}_q$ are the upper and lower bounds of nominal quantile level, respectively.

Figure 1 shows the load forecasts at a certain moment in different forms. The quantile prediction can either discretely approximate the probability distribution density P_t or be selected to build the prediction interval I_t.

III. Short-term Probabilistic Residential Load Forecasting Scheme

A. Overall Framework

The proposed scheme consists of the attention-based encoder-decoder network, adversarial domain adaptation network, LSTM RNNs, and quantile loss. Figure 2 depicts the overall framework of the proposed scheme. First, this framework takes the data of the new residential load, regular loads, and known factors as the samples. The data of the new residential load, i.e., $\left[\boldsymbol{o}^{\text{new}}_{t-t_w:t}, \boldsymbol{o}^{\text{known}}_{t-t_w:t+t_g+t_h}\right]$, are divided into a testing set $\left\{\boldsymbol{x}^{\text{new}}_t, \boldsymbol{y}^{\text{new}}_t\right\}$ from the starting time t_0 to the current time t_c and a training set $\left\{\boldsymbol{x}^{\text{est}}_t, \boldsymbol{y}^{\text{est}}_t\right\}$ from the earliest time $t^{\text{reg}}_0 = \max\left(\boldsymbol{t}_{n=0:N^{\text{reg}}}\right)$ to the current time t_c, where N^{reg} is the number of the historical periods of regular loads.

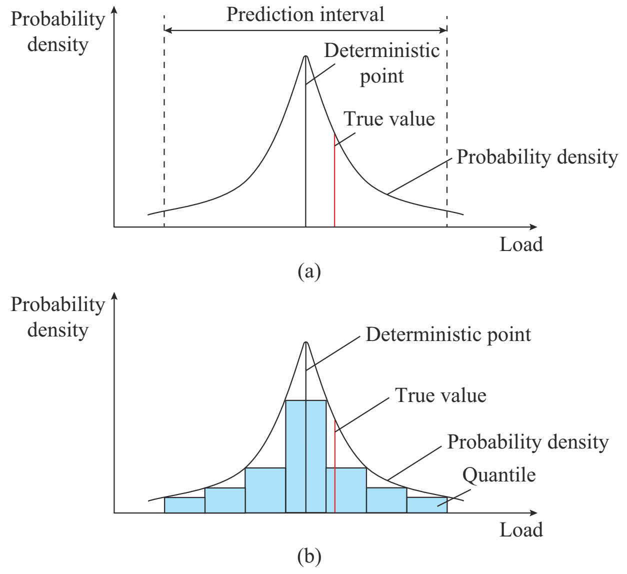


Fig. 1. Load forecasts at a specific moment in different forms. (a) Point, prediction interval, and probability density. (b) Point, quantile, and probability density.

The training set is also mixed with the data record related to regular loads and their known factors, i.e., $\left[\boldsymbol{o}^{\text{reg}}_{t-t_w:t}, \boldsymbol{o}^{\text{known}}_{t-t_w:t+t_g+t_h}\right]$.

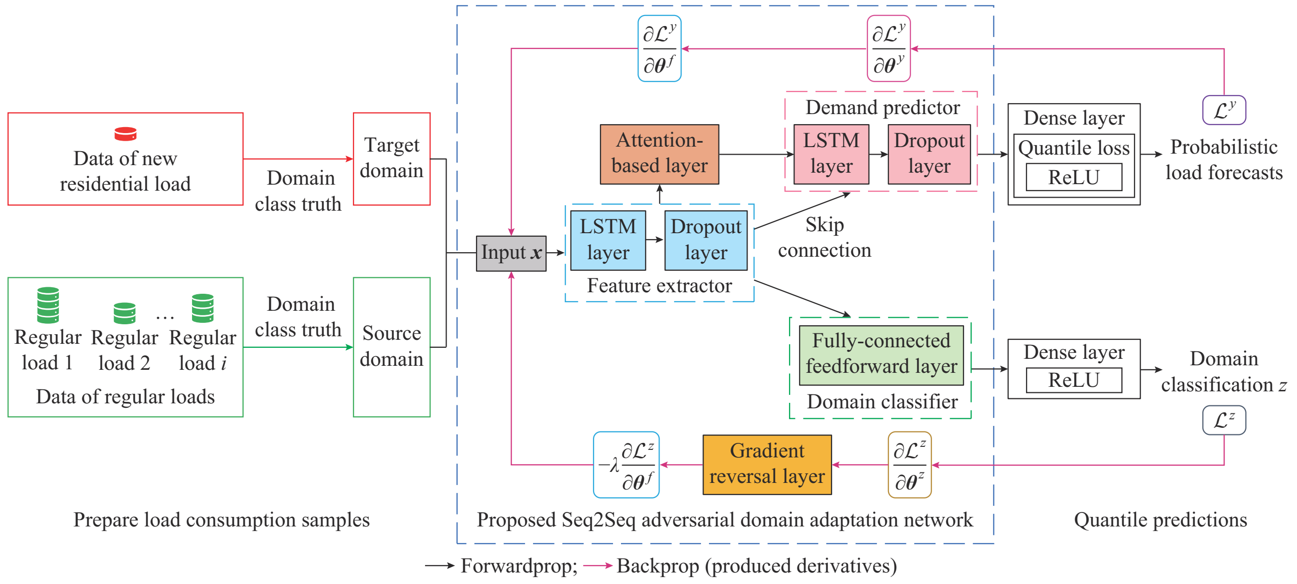

Data of new residential load
Domain class truth
Target domain
Regular load 1
Regular load 2
...
Regular load i
Data of regular loads
Domain class truth
Source domain
Prepare load consumption samples
$\frac{\partial \mathcal{L}^y}{\partial \theta^f}$
$\frac{\partial \mathcal{L}^y}{\partial \theta^y}$
Demand predictor
Attention-based layer
LSTM layer
Dropout layer
Input x
LSTM layer
Dropout layer
Feature extractor
Skip connection
Fully-connected feedforward layer
Domain classifier
$-\lambda\frac{\partial \mathcal{L}^z}{\partial \theta^f}$
Gradient reversal layer
$\frac{\partial \mathcal{L}^z}{\partial \theta^z}$
Proposed Seq2Seq adversarial domain adaptation network
Dense layer
Quantile loss
ReLU
$\mathcal{L}^y$
Probabilistic load forecasts
Dense layer
ReLU
Domain classification z
$\mathcal{L}^z$
Quantile predictions
Forwardprop; Backprop (produced derivatives)

Fig. 2. Overall framework of proposed scheme.

Generally, sufficient samples are required to identify flexible NNs with numerous parameters. The Seq2Seq adversarial domain adaptation network is proposed based on a feature extractor, a demand predictor, and a domain classifier by realizing a joint training process with the gradient reversal layer. For implementation, we adopt the generic elements as the submodules of the adversarial domain adaptation network including: ① the feature extractor by connecting an LSTM layer and a dropout layer, ② the demand predictor by connecting another LSTM layer and dropout layer, ③ a fully-connected feedforward layer as the domain classifier, and ④ dense layers after the demand predictor and domain classifier. In addition, we use an attention-based layer and a skip connection to capture the longer temporal dependency while mitigating the vanishing gradient problem, enhancing feature reuse, and facilitating the learning of identity mappings, which is motivated by the Seq2Seq RNN.

Both the data of the new residential load and regular loads can be leveraged via the Seq2Seq adversarial domain adaptation network, where the Seq2Seq RNN is well-trained by quantile loss before generating the multi-step probabilistic load forecasts. The optimal parameter assignments of the Seq2Seq adversarial domain adaptation network are represented by (11) and (12), and then the estimated network generates the final load forecasts by (13).

$$\left\{\boldsymbol{\theta}^{*,f},\boldsymbol{\theta}^{*,y},\boldsymbol{\theta}^{*,z}\right\}=\underset{\boldsymbol{\theta}^{f},\boldsymbol{\theta}^{y},\boldsymbol{\theta}^{z}}{\arg\min}\,R^{\text{emp}}\left(\boldsymbol{\theta}^{(\cdot)}|D^{\text{est}}\right) \tag{11}$$

$$\boldsymbol{\theta}^{*,\text{dense}}=\underset{\boldsymbol{\theta}^{\text{dense}}}{\arg\min}\,R^{\text{emp}}\left(\boldsymbol{\theta}^{\text{dense}}|D^{\text{new}},\boldsymbol{\theta}^{*,f},\boldsymbol{\theta}^{*,y},\boldsymbol{\theta}^{*,z}\right) \tag{12}$$

$$\hat{\boldsymbol{o}}_t^{\text{new}}=f^{\text{proposed}}\left(\boldsymbol{o}_{t-t_w:t}^{\text{new}},\boldsymbol{o}_{t-t_w:t+t_g+t_h}^{\text{known}};\boldsymbol{\theta}^{*,\text{proposed}}\right)\triangleq G^{\text{dense}}\left(G^{y}\left(G^{f}\left(\boldsymbol{x}_t^{\text{new}};\boldsymbol{\theta}^{*,f}\right);\boldsymbol{\theta}^{*,y}\right);\boldsymbol{\theta}^{*,\text{dense}}\right) \tag{13}$$

where D^{new} is the dataset of the new load; $\boldsymbol{\theta}^{*,\text{proposed}}$ is the vector of parameters for the proposed scheme; $\hat{\boldsymbol{o}}_t^{\text{new}}$ is the prediction of the new load; $f^{\text{proposed}}(\cdot)$ is the function of the proposed Seq2Seq adversarial domain adaptation network; $G^f(\cdot)$ and $\boldsymbol{\theta}^f$ are the feature extractor and the vector of its parameters, respectively; $G^y(\cdot)$ and $\boldsymbol{\theta}^y$ are the demand predictor and the vector of its parameters, respectively; $G^{\text{dense}}(\cdot)$ and $\boldsymbol{\theta}^{\text{dense}}$ are the output layer and the vector of its parameters, respectively; $\boldsymbol{\theta}^z$ is the vector of parameters for domain classfier; and $\boldsymbol{\theta}^{*,f}$, $\boldsymbol{\theta}^{*,y}$, $\boldsymbol{\theta}^{*,z}$, and $\boldsymbol{\theta}^{*,\text{dense}}$ are the optimal parameter assignments of $\boldsymbol{\theta}^f$, $\boldsymbol{\theta}^y$, $\boldsymbol{\theta}^z$, and $\boldsymbol{\theta}^{\text{dense}}$, respectively.

B. Adversarial Domain Adaptation Network

The parameters of the NN-based forecasting models are optimized by assuming the training data are independent and identically distributed. However, the accuracy of load forecasts could not be assured when the data distributions of the training and testing datasets vary, which is also known as a shift between data distributions of the training and test datasets. The concept of domain adaptation aims to learn a discriminative classifier or another predictor when there is a shift between data distributions of the training and test datasets, which is generally operated by matching the feature distributions in the source and target domains of synthetic or semi-synthetic image data. A dominant approach is to accomplish a feature space transformation that measures the similarity or dissimilarity between different distributions and maps the distributions of the source domain to the target domain [25]. In addition, the domain adaptation in the NN has been proposed to learn the features that are both discriminative for the target learning task in the source domain and invariant to the shift between the domains [26]. The combination of domain adaptation and neural architectures can be achieved by jointly optimizing the two components (i.e., demand predictor and domain classifier) and their underlying features in the training processes of the feedforward neural architecture models.

On the one hand, the demand predictor predicts the class label of domains during the training and testing processes. On the other hand, during the training process, the domain classifier discriminates between the source and the target domains. The feature extractor that is connected to the demand predictor and the domain classifier learns the deep features with discriminative and domain-invariance capabilities. Specifically, the parameters of the two components are optimized to minimize their error on the training set, and the parameters of the feature extractor are optimized to minimize the losses of the demand predictor and the domain classifier. After the optimization, the feature extractor learned for the source domain can be implemented in the target domain. The error of the joint training processes of the feature extractor, demand predictor, and domain classifier on the adversarial domain adaptation network is calculated by:

$$R^{\text{emp}}\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^y,\boldsymbol{\theta}^z\right)=\sum_{t=1,2,\dots,T}\mathcal{L}_t^y\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^y\right)-\lambda\sum_{t=1,2,\dots,T}\mathcal{L}_t^z\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^z\right)\triangleq \sum_{\substack{t=1,2,\dots,T\\ z_t=0}}\mathcal{L}^y\left(G^y\left(G^f\left(\boldsymbol{X}_t;\boldsymbol{\theta}^f\right);\boldsymbol{\theta}^y\right),\boldsymbol{y}_t\right)-\lambda\sum_{\substack{t=1,2,\dots,T\\ z_t=0}}\mathcal{L}^z\left(G^z\left(G^f\left(\boldsymbol{X}_t;\boldsymbol{\theta}^f\right);\boldsymbol{\theta}^z\right),z_t\right) \tag{14}$$

where $G^z(\cdot)$ is the domain classifier; T is the number of samples; z_t is the domain classification of observation, $z_t=0$ when the data at the time t belong to the target domain, $z_t=1$ when the data at the time t belong to the source domain; $\mathcal{L}_t^y(\cdot)$ and $\mathcal{L}_t^z(\cdot)$ are the loss functions; and λ is the trade-off weight between the losses of domain classifier G^z and demand predictor G^y.

As a result, the deep feature from the feature extractor G^f represents a space transformation between the distribution of the output of the demand predictor G^y in the source domain $\boldsymbol{X}_t^{\text{sour}}\sim\mathcal{D}^{\text{sour}}$ and that in the target domain $\boldsymbol{X}_t^{\text{tar}}\sim\mathcal{D}^{\text{tar}}$, i.e.,

$$\begin{cases} G^y\left(G^f\left(\boldsymbol{X}_t^{\text{sour}};\boldsymbol{\theta}^f\right);\boldsymbol{\theta}^y\right)\approx G^y\left(G^f\left(\boldsymbol{X}_t^{\text{tar}};\boldsymbol{\theta}^f\right);\boldsymbol{\theta}^y\right)\sim\mathcal{D}^y\\ \mathcal{D}^{\text{sour}}\neq\mathcal{D}^{\text{tar}}\\ \boldsymbol{X}_t^{\text{sour}}\sim\mathcal{D}^{\text{sour}}\\ \boldsymbol{X}_t^{\text{tar}}\sim\mathcal{D}^{\text{tar}}\end{cases} \tag{15}$$

where $\mathcal{D}^y$ is the true distribution of the outputs $\boldsymbol{y}_t$, $t=1,2,\dots,T$.

Moreover, the parameters $\hat{\boldsymbol{\theta}}^f$, $\hat{\boldsymbol{\theta}}^y$, and $\hat{\boldsymbol{\theta}}^z$ are orderly optimized to deliver the saddle point of (14):

$$\begin{cases}\left(\hat{\boldsymbol{\theta}}^f,\hat{\boldsymbol{\theta}}^y\right)=\underset{\boldsymbol{\theta}^f,\boldsymbol{\theta}^y}{\arg\min}\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^y,\hat{\boldsymbol{\theta}}^z\right)\\ \hat{\boldsymbol{\theta}}^z=\underset{\boldsymbol{\theta}^z}{\arg\min}\left(\hat{\boldsymbol{\theta}}^f,\hat{\boldsymbol{\theta}}^y,\boldsymbol{\theta}^z\right)\end{cases} \tag{16}$$

Specifically, the saddle point (16) is a stationary point of stochastic gradient descent (SGD) updates for the feedforward network model composed of a feature extractor G^f, a domain classifier G^d, and a demand predictor G^y:

$$\boldsymbol{\theta}^f\leftarrow\boldsymbol{\theta}^f-\mu\left(\frac{\partial\mathcal{L}_t^y}{\partial\boldsymbol{\theta}^f}-\lambda\frac{\partial\mathcal{L}_t^z}{\partial\boldsymbol{\theta}^f}\right) \tag{17}$$

$$\boldsymbol{\theta}^y\leftarrow\boldsymbol{\theta}^y-\mu\frac{\partial\mathcal{L}_t^y}{\partial\boldsymbol{\theta}^y} \tag{18}$$

$$\boldsymbol{\theta}^d\leftarrow\boldsymbol{\theta}^d-\mu\frac{\partial\mathcal{L}_t^z}{\partial\boldsymbol{\theta}^z} \tag{19}$$

where μ is the learning rate, which can vary over time.

However, we note that the minimization of the objective function (14), i.e., $\min R^{\text{emp}}$, includes a minimization optimization of prediction $\min\mathcal{L}_t^y$ and a maximization of classification $-\min\lambda\mathcal{L}_t^z$, and thus a "pseudo-function gradient" reversal layer $L^{\lambda}(\boldsymbol{x})$ only with the hyperparameter λ proposed in [26] is implemented to transform certain updates (17)-(19) into the standard form of SGD. Specifically, we adopt an identity transformation as the gradient reversal layer during the forward propagation, as given in (20). Then, the specified layer passes the gradient from the subsequent neural layer to the preceding layer by multiplying $-\lambda$ during the backpropagation, which is computed by:

$$L^{\lambda}(\boldsymbol{x}) = \boldsymbol{x} \tag{20}$$

$$\frac{\mathrm{d}L^{\lambda}}{\mathrm{d}\boldsymbol{x}} = -\lambda \boldsymbol{I} \tag{21}$$

where L^{λ} is a gradient reversal layer; and $\boldsymbol{I}$ is an identity matrix.

Based on L^{λ}, the modification of the objective function (14) with standard SGD forward propagation and backpropagation can be obtained as:

$$\tilde{R}^{\mathrm{emp}}\left(\boldsymbol{\theta}^{f}, \boldsymbol{\theta}^{y}, \boldsymbol{\theta}^{z}\right) = \sum_{\substack{t=1,2,\ldots,T \\ z_t = 0}} \mathcal{L}_t^{y}\left(G^{y}\left(G^{f}\left(\boldsymbol{X}_t; \boldsymbol{\theta}^{f}\right); \boldsymbol{\theta}^{y}\right), \boldsymbol{y}_t\right) + \sum_{t=1,2,\ldots,T} \mathcal{L}_t^{z}\left(G^{z}\left(L^{\lambda}\left(G^{f}\left(\boldsymbol{X}_t; \boldsymbol{\theta}^{f}\right)\right); \boldsymbol{\theta}^{z}\right), z_t\right) \tag{22}$$

C. LSTM RNNs

The RNN is a promising approach for solving load forecasting problems because the record of energy consumption often exhibits temporal characteristics. RNNs proposed for processing sequential data (such as speech, multivariate time series, and text) can leverage the time interdependency in the chronological values through the ideas of parameter-sharing and graph-unrolling [9], [27]. RNNs combine the input at the moment t $\boldsymbol{x}_t^{\mathrm{RNN}}$ and the hidden state at the last moment $\boldsymbol{h}_{t-1}^{\mathrm{RNN}}$, and produce the current hidden state $\boldsymbol{h}_t^{\mathrm{RNN}}$, as described in (23). The output at the moment t $\boldsymbol{y}_t^{\mathrm{RNN}}$ is then computed with the hidden state $\boldsymbol{h}_{t-1}^{\mathrm{RNN}}$, as described in (24). In multilayer networks, the recurrent neural layer composed of RNN cells can be used as a specific classification of the hidden layer. The hidden state $\boldsymbol{h}_t^{\mathrm{RNN}}$ will be different if the chronological order of the timestamps of the input $\boldsymbol{x}_t^{\mathrm{RNN}}$ changes.

$$\boldsymbol{h}_t^{\mathrm{RNN}} = \tanh\left(\boldsymbol{W}^{\mathrm{RNN}} \boldsymbol{h}_{t-1}^{\mathrm{RNN}} + \boldsymbol{U}^{\mathrm{RNN}} \boldsymbol{x}_t^{\mathrm{RNN}} + \boldsymbol{b}_2^{\mathrm{RNN}}\right) \tag{23}$$

$$\boldsymbol{y}_t^{\mathrm{RNN}} = f^{\mathrm{RNN}}\left(\boldsymbol{V}^{\mathrm{RNN}} \boldsymbol{h}_t^{\mathrm{RNN}} + \boldsymbol{b}_1^{\mathrm{RNN}}\right) \tag{24}$$

where $\tanh(\cdot)$ is a hyperbolic tanh function; $\boldsymbol{W}^{\mathrm{RNN}}, \boldsymbol{U}^{\mathrm{RNN}}$, and $\boldsymbol{V}^{\mathrm{RNN}}$ are the matrices of weights; and $\boldsymbol{b}_1^{\mathrm{RNN}}$ and $\boldsymbol{b}_2^{\mathrm{RNN}}$ are the vectors of biases.

The LSTM RNN adopts the gating mechanism to alleviate the gradient vanishing problem in conventional RNNs for modeling relatively long short-term dependency. It creates the path whose derivatives neither vanish nor explode to use the early temporal dependency (the hidden state) over the connection weights. The gradients flow via self-loops, where the weights are conditioned to the given data, as depicted in Fig. 3.

Specifically, the LSTM RNN takes an outer recurrence as the conventional RNN, an outer recurrence as a generic RNN, and an internal recurrence as the LSTM cell, as given in (25)-(30). In the outer recurrence, the element-wise mediate variable $\boldsymbol{z}_t^{\mathrm{LSTM}}$ correlates to the affine transformation of the input variable $\boldsymbol{x}_t^{\mathrm{LSTM}}$ and the hidden layer vector $\boldsymbol{h}_{t-1}^{\mathrm{LSTM}}$ from the last moment. In the internal recurrence, the internal state vector at the given time t $\boldsymbol{s}_t^{\mathrm{LSTM}}$ depends on the variable $\boldsymbol{z}_t^{\mathrm{LSTM}}$ that relates to the input gate $\boldsymbol{g}_t^{\mathrm{in}}$ and the variable $\boldsymbol{z}_{t-1}^{\mathrm{LSTM}}$ from the last moment that relates to the forget gate $\boldsymbol{g}_t^{\mathrm{forget}}$. The LSTM cell $\boldsymbol{y}_t^{\mathrm{LSTM}}$, i.e., the hidden layer vector $\boldsymbol{h}_t^{\mathrm{LSTM}}$, derives from the signal from the output gate $\boldsymbol{g}_t^{\mathrm{out}}$. This process has been depicted in [28].

$$\boldsymbol{z}_t^{\mathrm{LSTM}} = \tanh\left(\boldsymbol{W}^{\mathrm{LSTM}} \boldsymbol{h}_{t-1}^{\mathrm{LSTM}} + \boldsymbol{U}^{\mathrm{LSTM}} \boldsymbol{x}_t^{\mathrm{LSTM}} + \boldsymbol{b}^{\mathrm{LSTM}}\right) \tag{25}$$

$$\boldsymbol{s}_t^{\mathrm{LSTM}} = \boldsymbol{z}_t^{\mathrm{LSTM}} \boldsymbol{g}_t^{\mathrm{in}} + \boldsymbol{z}_{t-1}^{\mathrm{LSTM}} \boldsymbol{g}_t^{\mathrm{forget}} \tag{26}$$

$$\boldsymbol{y}_t^{\mathrm{LSTM}} = \boldsymbol{h}_t^{\mathrm{LSTM}} = \tanh\left(\boldsymbol{s}_t^{\mathrm{LSTM}}\right) \boldsymbol{g}_t^{\mathrm{out}} \tag{27}$$

$$\boldsymbol{g}_t^{\mathrm{in}} = \sigma\left(\boldsymbol{W}^{\mathrm{in}} \boldsymbol{h}_{t-1}^{\mathrm{LSTM}} + \boldsymbol{U}^{\mathrm{in}} \boldsymbol{x}_t^{\mathrm{LSTM}} + \boldsymbol{b}^{\mathrm{in}}\right) \tag{28}$$

$$\boldsymbol{g}_t^{\mathrm{forget}} = \sigma\left(\boldsymbol{W}^{\mathrm{forget}} \boldsymbol{h}_{t-1}^{\mathrm{LSTM}} + \boldsymbol{U}^{\mathrm{forget}} \boldsymbol{x}_t^{\mathrm{LSTM}} + \boldsymbol{b}^{\mathrm{forget}}\right) \tag{29}$$

$$\boldsymbol{g}_t^{\mathrm{out}} = \sigma\left(\boldsymbol{W}^{\mathrm{out}} \boldsymbol{h}_{t-1}^{\mathrm{LSTM}} + \boldsymbol{U}^{\mathrm{out}} \boldsymbol{x}_t^{\mathrm{LSTM}} + \boldsymbol{b}^{\mathrm{out}}\right) \tag{30}$$

where $\sigma(\cdot)$ is the Sigmoid function, which is another activation function besides $\tanh(\cdot)$; and $\boldsymbol{W}^{(\cdot)}$, $\boldsymbol{U}^{(\cdot)}$, and $\boldsymbol{b}^{(\cdot)}$ are the specific parameters in the LSTM cell.

D. Attention-based Encoder-decoder Network

The Seq2Seq RNN exhibits superior capability in modeling temporal characteristics between the input and output sequences by leveraging the local context around the target. The Seq2Seq RNN was first designed and used in computer vision, speech recognition, and natural language processing [15]. Unlike conventional RNNs, it contains an encoder to process sequential data of the influential factors, a decoder to generate multi-step load forecasts, and possibly an attention vector to mark essential dependencies in the sequence, as discussed in [28].

Figure 4 illustrates the encoder-decoder network with mixed inputs, where the encoder and decoder networks are two individual LSTM RNNs. The post-known factors, i.e., observed factors $\boldsymbol{o}_{t-t_w:t}^{\mathrm{obs}}$ and known factors $\boldsymbol{o}_{t-t_w:t}^{\mathrm{known}}$, are the inputs of the encoder. The encoder receives the input $\boldsymbol{x}_{t-t_w:t}^{\mathrm{enc}}$ and extracts the local temporal context $\boldsymbol{c}^{\mathrm{enc}}$ from the hidden state of encoder $\boldsymbol{h}_t^{\mathrm{enc}}$ as the first hidden state of decoder. The decoder combines the local context $\boldsymbol{c}^{\mathrm{enc}}$, the ground truth input $\boldsymbol{x}_{t+t_g:t+t_g+t_h}^{\mathrm{dec}}$ of pre-known factors $\boldsymbol{o}_{t:t+t_g+t_h}^{\mathrm{known}}$, and the self-generated input $\hat{y}_{t+t_g:t+t_g+t_h}^{\mathrm{dec}}$ to obtain predictions $\hat{\boldsymbol{y}}_{t+t_g:t+t_g+t_h}^{\mathrm{dec}}$ related to the hidden states $\left\{\boldsymbol{h}_{t+t_g}^{\mathrm{dec}}, \boldsymbol{h}_{t+t_g+t_h/n_h}^{\mathrm{dec}}, \ldots, \boldsymbol{h}_{t+t_g+t_h}^{\mathrm{dec}}\right\}$ that can be evaluated via the targets $\boldsymbol{y}_{t+t_g:t+t_g+t_h}^{\mathrm{dec}}$ during the lead time. The green dotted line represents that the prediction at the last moment is the input of the next moment.

E. Bayesian Target Encoding Based Data Pre-processing

Existing works usually consider calendar-related information as categorial features such as the hours in a day, the days in a week, the months in a year, the distinction between holidays and non-holidays, the distinction between weekdays and weekends, and the distinction between varying seasons, which can be used as part of the inputs of the load forecasting model after encoding. The one-hot encoding approach is one of the most common encoding approaches for categorial features. Figure 5 shows a common example of the one-hot encoding approach on monthly calendar-related features. We can see that the number of newly supplemented 0-1 features could be massive when the original feature involves too many categories, e.g., from 1 to 12 dimensions.

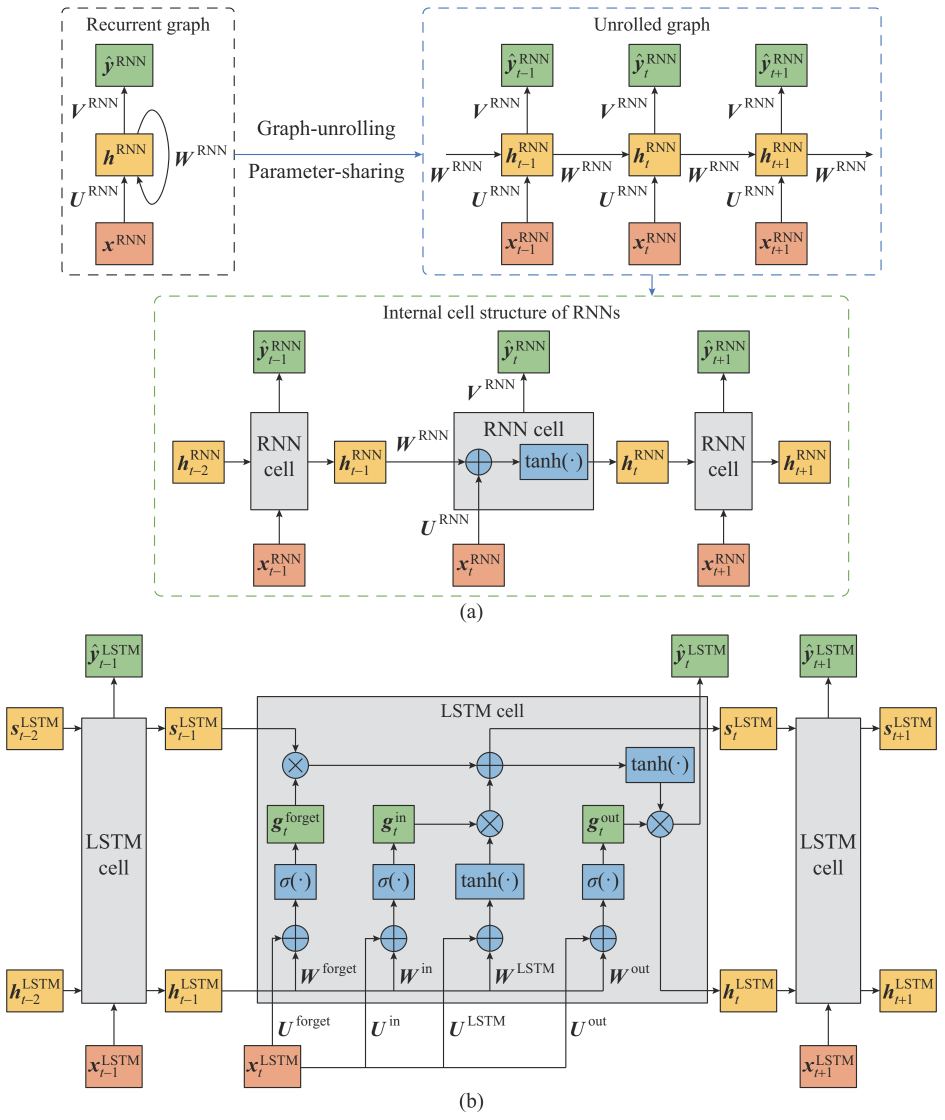


Fig. 3. Diagram of RNN variants. (a) Recurrent graph, unrolled graph, and internal cell structure of RNNs. (b) Internal cell structure of LSTM RNNs.

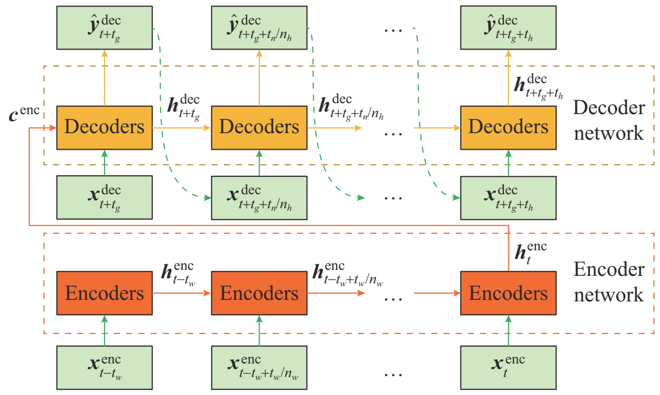
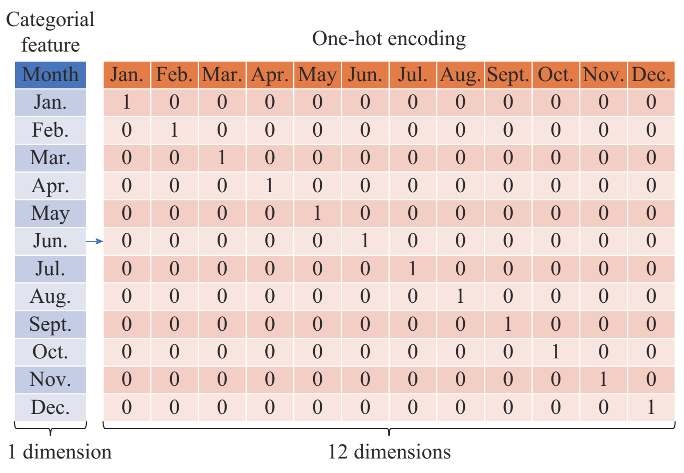


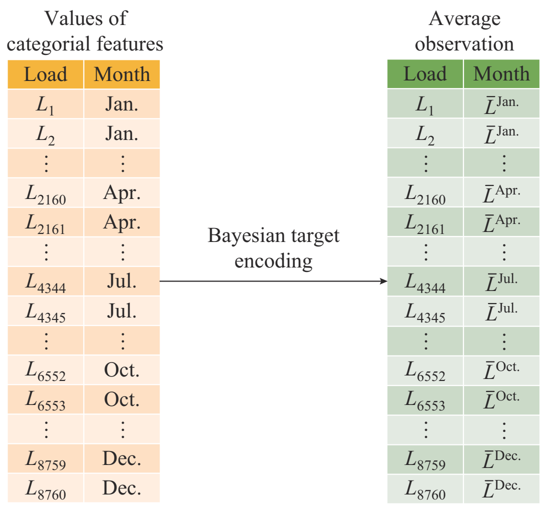
Fig. 4. Encoder-decoder network with mixed inputs.

To address the explosion of inputs by the one-hot encoding approach, we adopt the Bayesian target encoding to preprocess categorial features. Specifically, the values of the categorial features are compared with average observations of the load in the corresponding categorical values, as depicted in Fig. 6, thereby depicting the relationship between continuous and categorical features more explicitly, where L_1-L_{8760} are the hourly loads in a year; and $\bar{L}^{\text{Jan.}}$-$\bar{L}^{\text{Dec.}}$ are the average monthly loads from January to December.



Categorial feature	One-hot encoding											
Month	Jan.	Feb.	Mar.	Apr.	May	Jun.	Jul.	Aug.	Sept.	Oct.	Nov.	Dec.
Jan.	1	0	0	0	0	0	0	0	0	0	0	0
Feb.	0	1	0	0	0	0	0	0	0	0	0	0
Mar.	0	0	1	0	0	0	0	0	0	0	0	0
Apr.	0	0	0	1	0	0	0	0	0	0	0	0
May	0	0	0	0	1	0	0	0	0	0	0	0
Jun.	0	0	0	0	0	1	0	0	0	0	0	0
Jul.	0	0	0	0	0	0	1	0	0	0	0	0
Aug.	0	0	0	0	0	0	0	1	0	0	0	0
Sept.	0	0	0	0	0	0	0	0	1	0	0	0
Oct.	0	0	0	0	0	0	0	0	0	1	0	0
Nov.	0	0	0	0	0	0	0	0	0	0	1	0
Dec.	0	0	0	0	0	0	0	0	0	0	0	1
1 dimension	12 dimensions											

Fig. 5. Example of one-hot encoding approach on monthly calendar-related features.



Load	Month
L_1	Jan.
L_2	Jan.
⋮	⋮
L_{2160}	Apr.
L_{2161}	Apr.
⋮	⋮
L_{4344}	Jul.
L_{4345}	Jul.
⋮	⋮
L_{6552}	Oct.
L_{6553}	Oct.
⋮	⋮
L_{8759}	Dec.
L_{8760}	Dec.

Load	Month
L_1	$\bar{L}^{\text{Jan.}}$
L_2	$\bar{L}^{\text{Jan.}}$
⋮	⋮
L_{2160}	$\bar{L}^{\text{Apr.}}$
L_{2161}	$\bar{L}^{\text{Apr.}}$
⋮	⋮
L_{4344}	$\bar{L}^{\text{Jul.}}$
L_{4345}	$\bar{L}^{\text{Jul.}}$
⋮	⋮
L_{6552}	$\bar{L}^{\text{Oct.}}$
L_{6553}	$\bar{L}^{\text{Oct.}}$
⋮	⋮
L_{8759}	$\bar{L}^{\text{Dec.}}$
L_{8760}	$\bar{L}^{\text{Dec.}}$

Fig. 6. Example of Bayesian target encoding on monthly calendar features.

Moreover, we utilize a Z-score normalization method to map the values of the target load and its influential factors with different dimensions into a specific range [29]. The data $\tilde{y}_t$ calculated by (31) should follow the standard normal distribution, whose mean is 0 and standard deviation is 1.

$$\tilde{y}_t=\left(y_t-\frac{1}{T}\sum_{t=1}^{T}y_t\right)\left(\sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(y_t-\frac{1}{T}\sum_{t=1}^{T}y_t\right)^2}\right)^{-1} \tag{31}$$

F. Miscellanies

To evaluate the model performance, we compare load forecasts and true values via quantile score (QS) and Winkler score (WS) indexes.

The QS index calculated by (32) and (33) represents the mean of pinball losses throughout the lead time and all quantiles, respectively. A lower QS result indicates more precise forecasts compared with the ground truth values.

$$\mathcal{L}^{\text{qua}}_{\alpha_q,t}\left(G^y\left(G^f\left(\boldsymbol{X}_t;\boldsymbol{\theta}^f_{\alpha_q}\right);\boldsymbol{\theta}^y_{\alpha_q}\right),y_t\middle|\boldsymbol{\lambda}^*\right)\triangleq\mathcal{L}^{\text{qua}}_{\alpha_q,t}\left(\hat{y}_{\alpha_q,t},y_t\middle|\boldsymbol{\lambda}^*\right)=\begin{cases}q\left(\hat{y}_{\alpha_q,t}-y_t\right) & \hat{y}_{\alpha_q,t}\geq y_t\\ \left(q-1\right)\left(\hat{y}_{\alpha_q,t}-y_t\right) & \hat{y}_{\alpha_q,t}<y_t\end{cases} \tag{32}$$

$$R^{\text{qs}}\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^y\right)=\frac{1}{QT}\sum_{q=1}^{Q}\sum_{t=1}^{T}\mathcal{L}^{\text{qua}}_{\alpha_q,t}\left(G^y_{\boldsymbol{\theta}^y_{\alpha_q}}\left(G^f_{\boldsymbol{\theta}^f_{\alpha_q}}\left(\boldsymbol{X}_t\right)\right),y_t\middle|\boldsymbol{\lambda}^*\right) \tag{33}$$

where $\mathcal{L}^{\text{qua}}_{\alpha_q,t}(\cdot)$ is the quantile loss function for the nominal quantile level α_q at the time t; $\boldsymbol{\lambda}^*$ is the optimal hyperparameter; $\hat{y}_{\alpha_q,t}$ is the prediction for quantile α_q at the time t; and $R^{\text{qs}}(\cdot)$ is the average quantile loss.

Based on the quantile loss, the loss function of the dense layer after the feature extractor, the gradient update (with the impact on the loss $\nabla_{\boldsymbol{\theta}^{\text{dense}}}\mathcal{L}_{\alpha_q,t}$), and the optimization process is established by (34)-(36), respectively.

$$\mathcal{L}^{\text{dense}}_{\alpha_q,t}\left(\boldsymbol{\theta}^{\text{dense}}\right)=\sum\mathcal{L}^y_{\alpha_q,t}\left(\boldsymbol{\theta}^{\text{dense}};\hat{\boldsymbol{\theta}}^{*,f},\hat{\boldsymbol{\theta}}^{*,y},\hat{\boldsymbol{y}}^{\text{new}}_{\alpha_q,t},\boldsymbol{y}^{\text{new}}_t\right) \tag{34}$$

$$\boldsymbol{\theta}^{\text{dense}}=\boldsymbol{\theta}^{\text{dense}}-\mu\frac{\partial\mathcal{L}^{\text{dense}}_{\alpha_q,t}}{\partial\boldsymbol{\theta}^{\text{dense}}} \tag{35}$$

$$\hat{\boldsymbol{\theta}}^{\text{dense}}=\underset{\boldsymbol{\theta}^{\text{dense}}}{\arg\min}\,\mathcal{L}^{\text{dense}}_{\alpha_q,t}\left(\hat{\boldsymbol{\theta}}^{*,f},\hat{\boldsymbol{\theta}}^{*,y},\boldsymbol{\theta}^{\text{dense}}\right) \tag{36}$$

where $\mathcal{L}^{\text{dense}}_{\alpha_q,t}$ is the dense loss function for the nominal quantile level α_q at the time t.

The gradient updates (16). Besides, the loss function (22) with the loss $\nabla_{\boldsymbol{\theta}^f,\boldsymbol{\theta}^y,\boldsymbol{\theta}^z}\mathcal{L}^{\text{proposed}}_{\alpha_q,t}$ can be updated as:

$$\mathcal{L}^{\text{proposed}}_{\alpha_q,t}\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^y,\boldsymbol{\theta}^z\right)=\mathcal{L}^y_{\alpha_q,t}\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^y;\hat{\boldsymbol{y}}^{\text{est, load}}_{\alpha_q,t},\boldsymbol{y}^{\text{est, load}}_t\right)-\lambda\mathcal{L}^z_{\alpha_q,t}\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^z;\hat{\boldsymbol{z}}^{\text{est, label}}_t,\boldsymbol{z}^{\text{est, label}}_t\right) \tag{37}$$

$$\left(\hat{\boldsymbol{\theta}}^f,\hat{\boldsymbol{\theta}}^y\right)=\underset{\boldsymbol{\theta}^f,\boldsymbol{\theta}^y}{\arg\min}\,\mathcal{L}^y_{\alpha_q,t}\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^y,\hat{\boldsymbol{\theta}}^z\right) \tag{38}$$

$$\hat{\boldsymbol{\theta}}^z=\underset{\boldsymbol{\theta}^z}{\arg\min}\,\mathcal{L}^z_{\alpha_q,t}\left(\hat{\boldsymbol{\theta}}^f,\hat{\boldsymbol{\theta}}^y,\boldsymbol{\theta}^z\right) \tag{39}$$

where $\mathcal{L}^{\text{proposed}}_{\alpha_q,t}(\cdot)$ is the loss function of the proposed frame for the nominal quantile level α_q at the time t; $\hat{\boldsymbol{y}}^{\text{est, load}}_{\alpha_q,t}$ is the prediction of the new load for the nominal quantile level α_q at the time t; $\boldsymbol{y}^{\text{est, load}}_t$ is the ground truth of load at the time t; $\hat{\boldsymbol{z}}^{\text{est, label}}_t$ is the prediction of the domain label at the time t; and $\boldsymbol{z}^{\text{est, label}}_t$ is the ground truth label at the time t.

The WS index evaluates the sharpness and reliability of the prediction intervals constrained to the quantile bounds by (40) and (41). A lower WS is generally desired because an overly wide interval is meaningless [30].

$$\mathcal{L}^{\text{int}}_t\left(\hat{\boldsymbol{y}}_{\alpha_q,t},y_t\middle|\boldsymbol{\lambda}^*\right)=\begin{cases}\hat{y}_{\bar{\alpha}_q,t}-\hat{y}_{\underline{\alpha}_q,t} & \hat{y}_{\underline{\alpha}_q,t}\leq y_t\leq\hat{y}_{\bar{\alpha}_q,t}\\ \hat{y}_{\bar{\alpha}_q,t}+\left(\dfrac{2}{\beta}-1\right)\hat{y}_{\underline{\alpha}_q,t}-\dfrac{2}{\beta}y_t & y_t<\hat{y}_{\underline{\alpha}_q,t}\\ \left(1-\dfrac{2}{\beta}\right)\hat{y}_{\bar{\alpha}_q,t}-\hat{y}_{\underline{\alpha}_q,t}+\dfrac{2}{\beta}y_t & y_t>\hat{y}_{\bar{\alpha}_q,t}\end{cases} \tag{40}$$

$$R^{\text{ws}}\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^y\right)=\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}^{\text{int}}_t\left(\hat{\boldsymbol{y}}_{\alpha_q,t},y_t\middle|\boldsymbol{\lambda}^*\right) \tag{41}$$

$$\begin{cases}\bar{\alpha}_q-\underline{\alpha}_q=\beta\\ \underline{\alpha}_q=1-\bar{\alpha}_q=\dfrac{\beta}{2}\end{cases} \tag{42}$$

where $\mathcal{L}^{\text{int}}_t(\cdot)$ is the interval loss function at time t; $R^{\text{ws}}(\cdot)$ is the average WS loss; and β is the confidence level relating to $\bar{\alpha}_q$ and $\underline{\alpha}_q$.

In addition, the categorical cross-entropy (CCE) index, which is popular for classification problems, is adopted in (43)-(46) to compute the domain classification error. The ratio between the scales of samples from the source domain and the target domain γ is considered. The criterion makes the gradients computable when training the classification model. A lower CCE index $E^{\text{CCE}}\left(\boldsymbol{\theta}^f,\boldsymbol{\theta}^z\right)$ generally represents better classification capability. However, we require a relatively high classification error to achieve the distributional consistency of the output of the demand predictor G^y from source domain inputs $G^{y}_{\theta^{y}}\left(G^{f}_{\theta^{f}}\left(\boldsymbol{X}^{\text{sour}}_{t}\right)\right)$ and target domain inputs $G^{y}_{\theta^{y}}\left(G^{f}_{\theta^{f}}\left(\boldsymbol{X}^{\text{tar}}_{t}\right)\right)$, across the connection between the feature extractor G^{f}, the gradient reversal layer $L^{\lambda}(x)$, and the domain classifier G^{z}, i.e., $G^{z}_{\theta^{z}}\left(L^{\lambda}\left(G^{f}_{\theta^{f}}\left(\boldsymbol{X}_{t}\right)\right)\right)$, in this study.

$$\mathcal{L}^{\text{CCE}}_{t}\left(\text{Softmax}\left(G^{z}_{\theta^{z}}\left(L^{\lambda}\left(G^{f}_{\theta^{f}}\left(\boldsymbol{X}_{t}\right)\right)\right)\right),z_{t}\middle|\boldsymbol{\lambda}^{*}\right)\triangleq\mathcal{L}^{\text{CCE}}_{t}\left(\hat{z}_{t},z_{t}\middle|\boldsymbol{\lambda}^{*}\right)= -\left[\hat{z}_{t}\ln\left(p_{t}\right)+\gamma\left(1-\hat{z}_{t}\right)\ln\left(1-p_{t}\right)\right] \tag{43}$$

$$\text{Softmax}\left(z_{t}\right)=\frac{\mathrm{e}^{z_{t}}}{\sum_{t=1}^{T}\mathrm{e}^{z_{t}}} \tag{44}$$

$$\gamma=\frac{T^{\text{sour}}}{T^{\text{tar}}} \tag{45}$$

$$R^{\text{CCE}}\left(\boldsymbol{\theta}^{f},\boldsymbol{\theta}^{z}\right)=\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}^{\text{CCE}}_{t}\left(\text{Softmax}\left(G^{z}_{\theta^{z}}\left(L^{\lambda}\left(G^{f}_{\theta^{f}}\left(\boldsymbol{X}_{t}\right)\right)\right)\right),z_{t}\middle|\boldsymbol{\lambda}^{*}\right) \tag{46}$$

where $\mathcal{L}^{\text{CCE}}_{t}(\cdot)$ is the CCE loss function at the time t; $R^{\text{CCE}}(\cdot)$ is the average CCE loss; T^{sour} is the number of the source domains; T^{tar} is the number of target domains; p_{t} is the probability of the sample from the target domain at the time t; and $\text{Softmax}(\cdot)$ represents the normalized exponential function.

IV. Case Study

A. Dataset Description

A widely accepted real-life dataset published in [31] is utilized in this subsection. Specifically, the dataset contains hourly power consumption data of the entire New England system and eight parallel load zones including Rhode Island, New Hampshire, Northeast Massachusetts and Boston, West/Central Massachusetts, Southeast Massachusetts, Vermont, Maine, and Connecticut from January 2022 to March 2023. To implement the proposed scheme, we select two varying loads as the source domain with massive records of historical/regular loads and the target domain with limited data of new residential loads, and simulate diverse limited degrees of the source and target domains via the availability of corresponding records. For the whole dataset, the data are split into the training set (the samples before those in the validation and testing sets), the validation set (the next 672 samples before the testing set), and the testing set (the last 240 samples). When a certain proportion of the whole dataset is available, the same proportion of the whole training, validation, and testing sets would be used for simulation. Before training the model, we shuffle the order of the samples in the training, validation, and testing sets.

Generally, the similarity should be evaluated before adapting the knowledge in the source domain to the target domain. As a preliminary similarity assessment, we pick the New Hampshire load zone as the target domain and other load zones as potential source domains. A dominant *K*-shape clustering algorithm is used to process the eight data sources of energy consumption and calculate the similarity between them [32]. Table I exhibits the assessment results generated by monthly and half-year data, which are evaluated by the shape-based distance (SBD) index covering three main clusters, one of which includes load zones of Maine, Connecticut, Northeast Massachusetts and Boston, New Hampshire, Southeast Massachusetts, and Rhode Island. Thus, we assume New Hampshire as the target domain and select the most similar zones, Massachusetts and Boston, as the source domain for case studies. We note that the *K*-shape clustering algorithm can be used in other cases to identify the correlated source domains for the target domain.

TABLE I
Assessment Results Using *K*-shape Clustering Algorithm

Load zones	Monthly		Half-year	
	SBD	No. of clusters	SBD	No. of clusters
West/Central Massachusetts	0.01693278	0	0.01404867	0
Vermont	0.01923366	1	0.02019146	1
Rhode Island	0.00092255	2	0.00131910	2
New Hampshire	0	2	0	2
Northeast Massachusetts and Boston	0.00065996	2	0.00102957	2
Southeast Massachusetts	0.00141926	2	0.00289324	2
Maine	0.00068811	2	0.00220722	2
Connecticut	0.00082693	2	0.00192924	2

B. Experimental Settings

Before training the customized Seq2Seq RNN in adversarial domain adaptation, we manually determine the assignments of special parameters (i.e., hyperparameters) and tune them within specific ranges and numerical sets. This process involves experimental settings. In this study, these experimental settings are mainly divided into two groups: ① the training settings for the Seq2Seq RNN in adversarial domain adaptation in the proposed scheme, ablative analysis, and comparative models; and ② the model settings for the customized RNN itself. The experimental settings are summarized in Table II. The proposed scheme in the case study generates the 5%, 50%, and 95% quantile forecasts, which are evaluated by the QS and WS indexes. According to the results of the preliminary test, the target coverage level of 90% is enough for evaluating the performance of prediction intervals, and other quantiles such as 10%, 20%, 60%, and 70% can be covered by the interval between the 5% and 90% quantile forecasts.

We identify the appropriate assignments of the model settings (e.g., the learning rate, the batch size, and the dropout rate [33]) by repeating random searching procedures [34] for 80 times among $3\times3\times5$ groups of possible assignments. The numbers in bold in Table II are the identified results after 80 searches. Besides, we explore the assignments of *EL*, *EN*, *DL*, and *DN* through the sensitivity analysis to verify the stability and feasibility of the proposed scheme.

TABLE II
SUMMARY OF EXPERIMENTAL SETTINGS

Classification	Hyperparameter	Value
Training setting	Length of input sequence	24×1
	Length of output sequence	24×1
	Epoch number	10000
	Repetition number	10
	Early stopping patience	50
	Optimization algorithm	AdaM
	Number of random searching	50
Model setting	Learning rate	0.001, **0.0001**, 0.00001
	EL	1, 2, 3, 4, 5
	EN	10, 20, 40, 80, 160
	DL	1, 2, 3, 4, 5
	DN	10, 20, 40, 80, 160
	Batch size	**64**, 128, 256
	Activation function type	tanh(·)
	Dropout rate	0.1, **0.2**, 0.3, 0.4, 0.5

Note: *EL*, *EN*, *DL*, and *DN* are the numbers of encoder layers, encoder neurons per layer, decoder layers, and decoder neurons per layer, respectively.

With the empirically optimal experimental settings, we repeat the training process (i.e., optimizing model parameters) and the testing process (obtaining load forecasts and comparing them with ground truth values) for ten times to ensure the reproducibility and reliability of the results.

The case study is realized via the Python language (3.9.7), the TensorFlow wheel (2.7) [35], the Scikit-learn wheel (1.2.2) [36], the MAPIE wheel (0.6.4) [37], CUDA Toolkit platform (11.2.0), and the cuDNN library (8.1) through a machine of the Intel Core CPU (i7-11800H @ 2.30 Giga Hz), the RAM (16.0 Giga Bytes), and the NVIDIA GeForce GPU (RTX 3060 @ 8.0 Giga Bytes).

C. Sensitivity Analysis

To validate the feasibility of the proposed scheme, we conduct a detailed sensitivity analysis in terms of layers and neurons per layer in the encoder and decoder. Specifically, *EL* and *DL* are set to be 1, 2, 3, 4, and 5, and *EN* and *DN* range from 10 to 160. We then build varying Seq2Seq RNNs over the combinations of the two hyperparameter values. Similarly, the classification- and regression-related indexes are used to evaluate the model performance difference with varying assignments of hyperparameter. In this subsection, we assume that 100% of the source and target domains are available.

Table III shows the sensitivity analysis results of the proposed scheme with varying assignments of hyperparameters. From the average QS and the standard deviation for ten trials, the optimal numbers of *EL*, *EN*, *DL*, and *DN* should be 2, 40, 2, and 40, respectively, which have been used as the default configuration. Specifically, the lowest QS drops at the value of 0.00266. The increase of *EL*, *EN*, *DL*, and *DN* brings more flexibility to capture the time-series characteristic, resulting in the QS reduction from 0.01071 to 0.00266. Nevertheless, too much flexibility causes overfitting to the limited data, and the QS thus increases again when the number of neural layers is over 2 or the number of neurons per layer is over 40. In overfitting, the number of layers shows much more influence on the QS (from 0.00266 to 0.01071) than the number of neurons per layer (from 0.00266 to 0.00353), conforming to the consensus that the deeper network is more adaptive than the wider network.

TABLE III
SENSITIVITY ANALYSIS OF PROPOSED SCHEME WITH VARYING ASSIGNMENTS OF HYPERPARAMETERS

Hyperparameter				Average QS	Average WS
EL	*EN*	*DL*	*DN*		
1	40	2	40	0.00278 (0.00001)	0.02613 (0.0001)
2				0.00266 (0.00001)	0.02461 (0.0001)
3				0.00267 (0.00001)	0.02362 (0.0001)
4				0.00312 (0.00002)	0.02077 (0.0001)
5				0.01071 (0.00001)	0.06536 (0.0001)
2	10	2	40	0.00316 (0.00002)	0.02291 (0.0001)
	20			0.00267 (0.00001)	0.03284 (0.0002)
	80			0.00344 (0.00001)	0.02490 (0.0001)
	160			0.00353 (0.00001)	0.03123 (0.0002)
2	40	1	40	0.00316 (0.00002)	0.02869 (0.0003)
		3		0.00348 (0.00002)	0.02236 (0.0001)
		4		0.00501 (0.00004)	0.08236 (0.0007)
		5		0.00597 (0.00005)	0.07154 (0.0006)
2	40	2	10	0.00299 (0.00001)	0.03160 (0.0002)
			20	0.00269 (0.00001)	0.03320 (0.0002)
			80	0.00313 (0.00002)	0.02006 (0.0001)
			160	0.00344 (0.00002)	0.02474 (0.0001)

Note: the values in brackets are standard deviations.

Furthermore, the encoder exhibits more impact on the QS (from 0.00266 to 0.01071 in terms of *EL*, or from 0.00266 to 0.00353 in terms of *EN*) than the decoder (from 0.00266 to 0.00597 in terms of *DL* or from 0.00266 to 0.00344 in terms of *DN*). This empirically confirms that the encoder of the feature extractor G^f plays an essential role in the proposed scheme. Similarly, it should be considered that the optimal assignments for the lowest QS could cause narrow prediction intervals because of the high accuracy of the quantile forecasts and the ground truth values. Although the default configuration does not exhibit the best average WS ($\beta = 90\%$) and the standard deviation for ten trials, we recommend the suboptimal assignments of hyperparameters to balance QS and WS when implementing the proposed scheme.

D. Scenario Exploration

To vividly illustrate the performance of the proposed scheme in addressing the data lack phenomenon, we comprehensively simulate diverse scenarios related to the data availability of both source and target domains, and compare their performances. In other words, we evaluate the errors between ground truth values and prediction intervals as well as quantiles, when various proportions of samples from the two domains (10%, 20%, 40%, 60%, 80%, and 100%) are available, as summarized in Table IV. We use both regression-related criteria QS and WS ($\beta = 90\%$) and the classification index CCE to evaluate forecasting errors. It should be noted that *EL*, *EN*, *DL*, and *DN* are set to be 2, 40, 2, and 40, respectively.

TABLE IV
PERFORMANCE OF PROPOSED SCHEME

Proportion of samples (%)		Average CCE	Average QS	Average WS
Source domain	Target domain			
100	100	0.2500 (0.008)	0.00266 (0.00001)	0.02461 (0.0001)
80		0.2111 (0.006)	0.00331 (0.00002)	0.02306 (0.0001)
60		0.1687 (0.005)	0.00267 (0.00001)	0.03611 (0.0002)
40		0.1214 (0.004)	0.00351 (0.00003)	0.02442 (0.0001)
20		0.0667 (0.002)	0.00286 (0.00002)	0.03120 (0.0002)
10		0.0352 (0.001)	0.00319 (0.00002)	0.03129 (0.0002)
100	80	0.2361 (0.007)	0.00298 (0.00001)	0.02240 (0.0001)
	60	0.2187 (0.006)	0.00312 (0.00002)	0.02542 (0.0001)
	40	0.1964 (0.006)	0.00509 (0.00004)	0.02142 (0.0001)
	20	0.1667 (0.005)	0.00350 (0.00002)	0.02632 (0.0001)
	10	0.1477 (0.005)	0.00471 (0.00004)	0.02548 (0.0001)
80	80	0.2500 (0.001)	0.00338 (0.00002)	0.02741 (0.0002)
60	60	0.2500 (0.001)	0.00363 (0.00002)	0.02963 (0.0002)
40	40	0.2500 (0.001)	0.00402 (0.00003)	0.02753 (0.0001)
20	20	0.2500 (0.001)	0.00414 (0.00003)	0.02721 (0.0001)
10	10	0.2500 (0.001)	0.00862 (0.00006)	0.06526 (0.0003)

Note: the values in brackets are standard deviations.

From Table IV, the average CCE and its standard deviation for ten trials reduce as the gap between the proportions of samples from the source and target domains becomes larger, and the lowest value is 0.0352 when utilizing 10% samples from the source domain and 100% samples from the target domain. Meanwhile, the CCE approaches and maintains at 0.2500 when the proportions of the source and target domains become similar. On the other hand, we obtain the lowest average QS and its standard deviation when all samples are available, i. e., 100% samples from both the source and target domains are utilized. However, the prediction interval can be narrow and tight if probabilistic forecasts are close to ground truth values. In summary, the proposed scheme shows the ability to extract deep features that transfer knowledge from the source domain to the target domain and decreases the QS from 0.00862 (when 10% samples from the source domain are utilized) to 0.00471 (when 100% samples from the source domain are utilized).

Moreover, Table IV confirms the adversarial manner in the proposed scheme. The implicit features from the feature extractor G^f make the records of regular loads an advantage to the demand predictor G^y in generating the energy demand predictions for the new load while confusing the domain classifier G^z in judging which domain a specific sample is from. Specifically, the CCE index increases from 0.0352 to 0.2500, while the average QS decreases from 0.00319 to 0.00266 as the proportion of samples from the source domain grows from 10% to 100%.

E. Comparison of Proposed Scheme and Other Schemes

To prove the superiority of the proposed scheme, we compare probabilistic forecasts generated by machine learning and deep learning schemes. The machine learning schemes include random forests (RFs) and gradient boosting decision trees (GBDTs). Deep learning schemes include generic fully connected feedforward NN (gen-FFNN), residual FFNN (res-FFNN), gated recurrent unit (GRU) RNN, LSTM RNN, generic temporal convolutional network (gen-TCN), conditional TCN (con-TCN), and WaveNet. In addition, static and teacher force (TF) Seq2Seq RNNs without domain adaptation are applied as ablative models to compare with the proposed scheme. We also utilize the QS and WS indexes to evaluate the forecasting results obtained from these schemes with different proportions of samples from the two domains, as summarized in Table V. The settings of comparative schemes are determined according to [38]-[42].

TABLE V
FORECASTING RESULTS OF PROPOSED SCHEME COMPARED WITH OTHER SCHEMES

Scheme	100% samples from target domain		10% samples from target domain	
	QS	WS	QS	WS
GBDT	0.00253 (0.00001)	0.02946 (0.0003)	0.00666 (0.00001)	0.02603 (0.0001)
RF	0.00269 (0.00001)	0.07370 (0.0007)	0.00648 (0.00003)	0.05934 (0.0002)
Gen-FFNN	0.00317 (0.00002)	0.08682 (0.0008)	0.00636 (0.00003)	0.22743 (0.0011)
Res-FFNN	0.00317 (0.00002)	0.08645 (0.0008)	0.00623 (0.00003)	0.21977 (0.0010)
LSTM RNN	0.00266 (0.00001)	0.04991 (0.0005)	0.01715 (0.00060)	0.05719 (0.0003)
GRU RNN	0.00291 (0.00001)	0.05945 (0.0005)	0.01419 (0.00050)	0.06197 (0.0003)
Gen-TCN	0.01144 (0.00010)	0.03771 (0.0003)	0.03312 (0.00150)	0.13722 (0.0006)
Con-TCN	0.00775 (0.00007)	0.03559 (0.0003)	0.02412 (0.00100)	0.10000 (0.0004)
WaveNet	0.00845 (0.00007)	0.04233 (0.0004)	0.01082 (0.00050)	0.17805 (0.0100)
TF Seq2Seq RNN	0.00258 (0.00001)	0.01209 (0.0001)	0.09575 (0.00041)	0.43193 (0.0020)
Static Seq2Seq RNN	0.00323 (0.00002)	0.01173 (0.0001)	0.09575 (0.00040)	0.43193 (0.0020)
Proposed	0.00266 (0.00001)	0.02461 (0.0002)	0.00471 (0.00002)	0.02548 (0.0001)

Note: the values in brackets are standard deviations.

From Table V, the TF Seq2Seq RNN accomplishes the lowest average QS of 0.00258 with the standard deviation of 0.00001 for ten trials as expected, and the static Seq2Seq RNN reaches the lowest WS of 0.01173 when β is 90%, given 100% samples from the target domain. In this half, when the dataset is sufficient, the proposed scheme outperforms the most dominant schemes with the same QS as the LSTM RNN of 0.00266. However, the task becomes much more difficult when the scale of the samples is limited to only 10% of the entire dataset, which simulates potential situations of the new residential load. The limited data have resulted in the degradation of all comparative schemes. For example, the QS values of static and TF Seq2Seq RNNs without adversarial domain adaptation have degraded from 0.00258 and 0.00323 to 0.09575, respectively, and the WS values have reduced from 0.01209 and 0.01173 to 0.43193, respectively.

Meanwhile, the proposed scheme exhibits its superiority in leveraging sufficient records of regular loads and supplementing the available dataset when training the adaptive Seq2Seq RNN. Therefore, the proposed scheme keeps generating accurate forecasts and can accomplish the best performance in terms of both the QS index (0.00471) and the WS index (0.02548). Given the entire target domain, we validate the performance through the proposed scheme with true profiles by vividly illustrating a group of day-ahead quantile predictions and the ground truth values, as shown in Fig. 7, in which the 50% quantile forecasts fit the ground truth value well. The interval between 5% and 95% quantile forecasts also covers the ground truth values as expected. On the other hand, Fig. 8 depicts the profiles of the 5%, 50%, and 95% quantile forecasts and the target when only 10% samples of the target domain is available, further demonstrating the effective coverage of the ground truth values and the fitting capability.

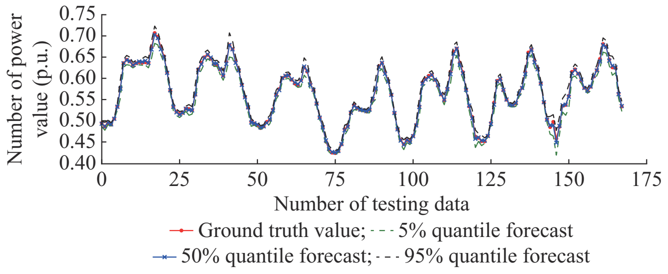


Fig. 7. Day-ahead quantile predictions and ground truth values given 100% samples from target domain.

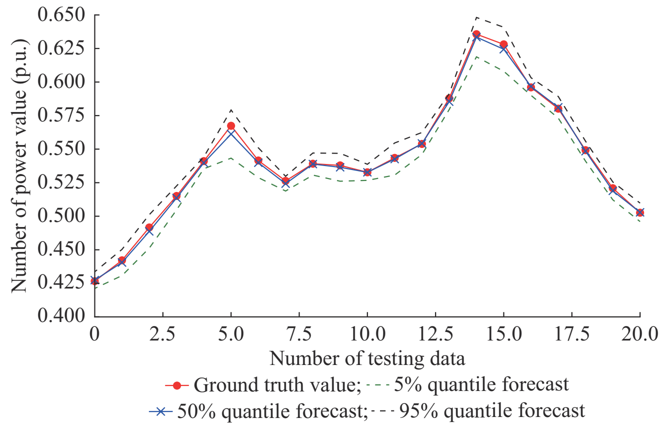


Fig. 8. Day-ahead quantile forecasts and ground truth values given 10% samples from target domain.

V. Conclusion

The proportion and scale of renewable power generation such as solar power in the distribution system keep increasing, so it is imperative to develop load forecasting technologies to obtain precise net load profiles for planning and dispatching the power system in the context of penetrating renewables. This paper focuses on the volatile residential load series and addresses the data lack problem as a significant branch in the field of probabilistic load forecasting. The proposed scheme included a Seq2Seq RNN over two LSTM layers as the feature extractor and the demand predictor, respectively, and a fully connected feedforward layer as the domain classifier.

To implement the adversarial domain adaptation network, we mix historical records and newly collected residential load observations, train the Seq2Seq adversarial domain adaptation network with samples from source and target domains, and generate accurate forecasts.

In the case study, we investigate the stability and feasibility of the proposed scheme for day-ahead probabilistic forecasting by limiting the scale of available data from the source or target domains. The results show that the methods or techniques widely accepted may lose their extraordinary capability and become vulnerable when data resources are inevitably limited or insufficient. Meanwhile, although the Seq2Seq RNN is often fed with massive data, the proposed scheme can maintain robust performance for precise load forecasts as we gradually reduce the available scales of the source and target domains. This finding can inspire further discussions and investigations of new technologies to deal with the data lack phenomenon in this area. Future work will consider the attention mechanism when integrating domain adaptation into a Seq2Seq RNN.

References

[1] W. Liao, S. Wang, B. Bak-Jensen *et al.*, "Ultra-short-term interval prediction of wind power based on graph neural network and improved bootstrap technique," *Journal of Modern Power Systems and Clean Energy*, vol. 11, no. 4, pp. 1100-1114, Jul. 2023..

[2] J. Zhu, H. Dong, W. Zheng *et al.*, "Review and prospect of data-driven techniques for load forecasting in integrated energy systems," *Applied Energy*, vol. 321, p. 119269, Sept. 2022.

[3] IEA. (2019, Dec.). Renewables 2019. [Online]. Available: https://www.iea.org/reports/renewables-2019/distributed-solar-pv

[4] IEA. (2021, Dec.). Renewables 2021. [Online]. Available: https://www.iea.org/reports/renewables-2021

[5] Q. Cui, J. Zhu, J. Shu *et al.*, "Comprehensive evaluation of electric power prediction models based on D-S evidence theory combined with multiple accuracy indicators," *Journal of Modern Power Systems and Clean Energy*, vol. 10, no. 3, pp. 597-605, May 2022.

[6] L. Ghelardoni, A. Ghio, and D. Anguita, "Energy load forecasting using empirical mode decomposition and support vector regression," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 549-556, Mar. 2013.

[7] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting – a novel pooling deep RNN," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271-5280, Sept. 2018.

[8] 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, Feb. 2001.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, May 2015.

[10] T. Elsken, J. H. Metzen, and F. Hutter. *Automated Machine Learning*. Cham: Springer, 2019: 63-77.

[11] B. Stephen, X. Tang, P. R. Harvey *et al.*, "Incorporating practice theory in sub-profile models for short term aggregated residential load forecasting," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1591-1598, Jul. 2017.

[12] W. Kong, Z. Y. Dong, D. J. Hill *et al.*, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1087-1088, Jan. 2018.

[13] J. Ponoćko and J. V. Milanović, "Forecasting demand flexibility of ag-

gregated residential load using smart meter data," *IEEE Transactions on Power Systems*, vol. 33, no. 5, pp. 5446-5455, Sept. 2018.
[14] W. Kong, Z. Y. Dong, Y. Jia *et al*., "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841-851, Jan. 2019.
[15] K. Cho, B. van Merrienboer, C. Gulcehre *et al*., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), Doha, Qatar, Oct. 2014, pp. 1724-1734.
[16] E. Skomski, J. Y. Lee, W. Kim *et al*., "Sequence-to-sequence neural networks for short-term electrical load forecasting in commercial office buildings," *Energy and Buildings*, vol. 226, p. 110350, Nov. 2020.
[17] N. Mughees, S. A. Mohsin, A. Mughees *et al*., "Deep sequence to sequence Bi-LSTM neural networks for day-ahead peak load forecasting," *Expert Systems with Applications*, vol. 175, p. 114844, Aug. 2021.
[18] Z. Masood, R. Gantassi, Ardiansyah *et al*., "A multi-step time-series clustering-based Seq2Seq LSTM learning for a single household electricity load forecasting," *Energies*, vol. 15, no. 7, p. 2623, Apr. 2022.
[19] M. Shepero, D. van der Meer, J. Munkhammar *et al*., "Residential probabilistic load forecasting: a method using Gaussian process designed for electric load data," *Applied Energy*, vol. 218, pp. 159-172, May 2018.
[20] L. Cheng, H. Zang, Y. Xu *et al*., "Probabilistic residential load forecasting based on micrometeorological data and customer consumption pattern," *IEEE Transactions on Power Systems*, vol. 36, no. 4, pp. 3762-3775, Jul. 2021.
[21] C. Li, Z. Dong, L. Ding *et al*., "Interpretable memristive LSTM network design for probabilistic residential load forecasting," *IEEE Transactions on Circuits and Systems I*: *Regular Papers*, vol. 69, no. 6, pp. 2297-2310, Jun. 2022.
[22] S. Li, Y. Zhong, and J. Lin, "AWS-DAIE: incremental ensemble short-term electricity load forecasting based on sample domain adaptation," *Sustainability*, vol. 14, no. 21, p. 14205, Oct. 2022.
[23] M. Huang and J. Yin, "Research on adversarial domain adaptation method and its application in power load forecasting," *Mathematics*, vol. 10, no. 18, p. 3223, Sept. 2022.
[24] J. Wang, X. Xiong, Z. Li *et al*., "Wind forecast-based probabilistic early warning method of wind swing discharge for OHTLs," *IEEE Transactions on Power Delivery*, vol. 31, no. 5, pp. 2169-2178, Oct. 2016.
[25] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell *et al*., "Unsupervised domain adaptation by domain invariant projection," in *Proceedings of 2013 IEEE International Conference on Computer Vision*, Sydney, Australia, Dec. 2013, pp. 769-776.
[26] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, Jul. 2015, pp. 1180-1189.
[27] F. He, J. Zhou, Z. Feng *et al*., "A hybrid short-term load forecasting model based on variational mode decomposition and long short-term memory networks considering relevant factors with Bayesian optimization algorithm," *Applied Energy*, vol. 237, pp. 103-116, Mar. 2019.
[28] H. Dong, J. Zhu, S. Li *et al*., "Short-term residential household reactive power forecasting considering active power demand via deep Transformer sequence-to-sequence networks," *Applied Energy*, vol. 329, p. 120281, Jan. 2023.
[29] Q. Cui, J. Zhu, J. Shu *et al*., "Comprehensive evaluation of electric power prediction models based on D-S evidence theory combined with multiple accuracy indicators," *Journal of Modern Power Systems and Clean Energy*, vol. 10, no. 3, pp. 597-605, May 2022.
[30] W. Zhang, H. Quan, O. Gandhi *et al*., "Improving probabilistic load forecasting using quantile regression NN with skip connections," *IEEE Transactions on Smart Grid*, vol. 11, no. 6, pp. 5442-5450, Nov. 2020.
[31] ISO New England Inc. (2022, Dec.). Energy, load, and demand reports. [Online]. Available: https://www.iso-ne.com/
[32] J. Paparrizos and L. Gravano, "*k*-shape: efficient and accurate clustering of time series," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, Melbourne, Australia, Jan. 2015, pp.1855-1870.
[33] N. Srivastava, G. Hinton, A. Krizhevsky *et al*., "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, Jan. 2014.
[34] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 2, Feb. 2012.
[35] M. Abadi, A. Agarwal, P. Barham *et al*. (2016, Mar.). TensorFlow: large-scale machine learning on heterogeneous distributed systems. [Online]. Available: https://arxiv.org/abs/1603.04467
[36] F. Pedregosa, G. Varoquaux, A. Gramfor *et al*., "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825-2830, Nov. 2011.
[37] V. Taquet, V. Blot, T. Morzadec *et al*., (2022, July). MAPIE: an open-source library for distribution-free uncertainty quantification. [Online]. Available: https://arxiv.org/abs/2207.12274
[38] A. Gasparin, S. Lukovic, and C. Alippi. (2019, Jul.). Deep learning for time series forecasting: the electric load case. [Online]. Available: https://arxiv.org/abs/1907.09207
[39] K. Chen, K. Chen, Q. Wang *et al*., "Short-term load forecasting with deep residual networks," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3943-3952, Jul. 2019.
[40] S. Bai, J. Z. Kolter, and V. Koltun. (2018, Mar.). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. [Online]. Available: https://arxiv.org/abs/1803.01271
[41] A. Borovykh, S. Bohte, and C. W. Oosterlee. (2017, Mar.). Conditional time series forecasting with convolutional neural networks. [Online]. Available: https://arxiv.org/abs/1703.04691
[42] A. van den Oord, S. Dieleman, H. Zen *et al*. (2016, Sept.). WaveNet: a generative model for raw audio. [Online]. Available: https://arxiv.org/abs/1609.03499

Hanjiang Dong received the B.S. degree in electrical engineering and its automation from Jinan University, Guangzhou, China, in 2020. He is currently working toward the Ph.D. degree with the School of Electric Power Engineering, South China University of Technology, Guangzhou, China, and the dual Ph. D. degree of the Department of Electrical Engineering, The Hong Kong Polytechnic University, Hong Kong, China. His current research interests include energy forecasting, microgrid, and robust optimization.

Jizhong Zhu received the B.S., M.S., and Ph.D. degrees in electrical engineering from Chongqing University, Chongqing, China, in 1985, 1987, and 1990, respectively. He is a Fellow of IEEE and a Professor at the School of Electric Power Engineering, South China University of Technology, Guangzhou, China. His research interests include power system operation and control, smart grid, microgrid, virtual power plant, electric vehicle, renewable energy application, and integrated smart energy system.

Shenglin Li received the B.S. and M.S. degrees from Shanghai University of Electric Power, Shanghai, China, in 2016 and 2019, respectively, and the Ph.D. degree in electrical engineering from South China University of Technology, Guangzhou, China, in 2023. He is currently working as a Post-doctoral Fellow in South China University of Technology, Guangzhou, China. His research interests include demand-side energy management, and optimal dispatch in microgrids and distribution networks.

Yuwang Miao received the B.E. degree in electrical engineering and automation from Xiamen University, Xiamen, China, in 2021. He is currently pursuing the M.S, degree of electrical engineering in South China University of Technology, Guangzhou, China. His research interests include load forecasting of power system based on artificial intelligence.

Chi Yung Chung received the B.Eng. (Hons.) and Ph.D. degrees in electrical engineering from The Hong Kong Polytechnic University, Hong Kong, China, in 1995 and 1999, respectively. He is currently the Chair Professor of Power Systems Engineering and the Head of the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University. He is the Vice Editor-in-Chief of Journal of Modern Power Systems and Clean Energy. His research interests include smart grid technology, renewable energy, power system stability and control, planning and operation, computational intelligence applications, electricity market, and electric vehicle charging.

Ziyu Chen received the B.S. degree in electrical engineering from Guangdong University of Technology, Guangzhou, China, in 2017, and the Ph.D. degree in electrical engineering from South China University of Technology, Guangzhou, China, in 2022. She is currently a Postdoctoral and Research Assistant in electrical engineering from South China University of Technology. Her current research interests include cyber-physical-social system, operation and control of smart energy system and cyber-attack detection.