Consumer Baseline Load Estimation in Demand Response: A Generative Adversarial Networks Approach

1st Zhenyi Wang
State Key Laboratory of Internet of Things for Smart City
University of Macau
Macao, China
zhenyi.wang@connect.um.edu.mo

2nd Hongcai Zhang
State Key Laboratory of Internet of Things for Smart City
University of Macau
Macao, China
hczhang@um.edu.mo

***Abstract*—Customer baseline load estimation plays an important role in the financial compensation settlement for incentive-based demand response. Current studies either rely on baseline load data from other consumers with similar load patterns or need the manual selection of valid features. However, these requirements are hard to satisfy in practice due to the fluctuation and personalization of customer power consumption profiles. To overcome this challenge, we propose a generative adversarial network-based customer baseline load estimation method. First, the estimation problem is converted into a time-series missing data problem. Then, the gated recurrent unit network is adopted to automatically distill the sequential information of historical baseline load data. Further, we take advantage of the generative adversarial network to recover incomplete baseline load data through a game between the generator and discriminator. Case studies based on a realistic dataset of building load demonstrate the effectiveness and superiority of the proposed method.**

***Index Terms*—Demand response, baseline load estimation, generative adversarial network, gated recurrent unit**

I. Introduction

With the continuous investment in renewable energy sources in the power system, it becomes increasingly difficult to maintain the balance between power supply and demand [1]. In order to guarantee the stability and reliability of the power system, more flexible resources need to be exploited to eliminate the uncertainty caused by renewable energy sources. As a major power regulation measure, demand response (DR) plays an important role in ensuring a stable and reliable power system [2].

DR changes customers' normal electricity consumption patterns by reducing or shifting their electricity demand to maintain a real-time power balance. Currently, DR is generally divided into two categories: 1) price-based DR, and 2) incentive-based DR [3]. For the price-based DR, the system operator issues time-varying tariffs to influence customers' normal electricity consumption schedules. For the incentive-based DR, the operator directly provides financial compensation to encourage customers to reduce their electricity demand during peak load periods. In order to determine the adequate financial compensation, it is necessary to accurately evaluate the actual amount of load reduction by each customer, which is the difference between the customer's nominal power consumption (i.e., the baseline load) in the absence of DR and the actual power consumption with DR. However, this is nontrivial because the consumer baseline load (CBL) cannot be recorded during the DR periods and it relies on estimation [4], [5]. An example of the comparison of CBL and actual meter load is shown in Figure 1. The red curve is the CBL and the blue curve is the actual load. In the non-DR periods, the two values are equal; while during the DR periods, the deviation between the two load curves is the customer's load reduction. However, when a DR event occurs, the CBL will no longer exist and cannot be measured. Therefore, as the basis for calculating demand response compensation, the accurate estimation of CBL is critical. Too high or too low estimation would affect the effective operation of DR [6].

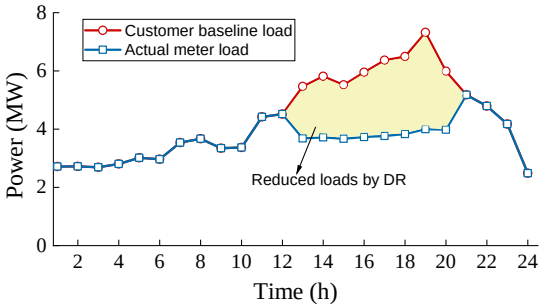

Fig. 1. The diagram of CBL.

In order to achieve the accurate estimation of CBLs, current studies are mainly divided into three categories: averaging, regression, and control group methods. The averaging method estimates the CBL based on the average of the customer's historical load [7]. Since customer load is susceptible to natural and social factors, resulting in high volatility, the

This paper is funded in part by the Science and Technology Development Fund, Macau SAR (File no. SKL-IOTSC(UM)-2021-2023, and File no. 0137/2019/A3) and in part by the Zhuhai science and technology innovation Bureau (File no. ZH22017002210022PWC).

accuracy of such methods is often poor. The regression method constructs models based on historical load, weather conditions, time series and other factors to estimate the CBL [8]–[10]. However, these features need to be extracted manually and their effectiveness is difficult to measure, so the robustness and generality of these methods cannot be guaranteed. For the control group method, all customers are divided into the DR group and non-DR group [4], [11]. Loads of the non-DR group are used to estimate the CBL of the DR group in the DR periods. This approach assumes that both groups of customers have similar electricity consumption patterns during the same periods, which may be not true in practice.

As mentioned above, the CBL does not exist and cannot be measured when the consumer is involved in DR. Therefore, we can treat the CBL during the DR period as missing data. Considering that electricity load is typically time-series data, we can then regard the CBL problem as a time-series missing data problem. Thus, estimating the CBL is equivalent to recovering the missing data. For the time-series missing data problem in the power system, there are some deep learning-based methods that have been studied recently. For example, Deng et al. [12] applied the long short-term memory (LSTM) for type recognition and time location of power quality disturbances. Zang et al. [13] proposed a novel method based on LSTM to forecast the day-ahead residential load. For the time-series data recovery problem, Jeong et al. [14] utilized the mixture factor analysis method to impute the missing values in building load data. However, there are few studies on time-series data recovery for baseline load estimation. Wang et al. [15] proposed a SAE-based method to recover the residential CBL, but the reconstruction performance is vulnerable due to the pseudo-load pool.

To overcome the aforementioned challenge, we propose a novel CBL estimation method based on the generative adversarial network (GAN) [16], which has recently been used for scene generation in the power system [17]. The main contributions of this paper are summarized as follows:

1) A GAN-based data recovery approach is proposed for CBL estimation. This method transforms the CBL estimation problem into a time-series data recovery issue. Through the complete data generated by GAN, the CBL with missing data in the DR period is recovered, which is equivalent to realizing CBL estimation.
2) The gated recurrent unit network is deployed in GAN as the base model. The network extracts the temporal relationships of the CBL without manual operation, which enables the proposed method to have high robustness.

The remainder of this paper is organized as follows. In Section II, the framework and details of the proposed methodology are elaborated. The effectiveness of the proposed method is validated by numerical experiments in Section III. Finally, Section IV concludes this paper.

II. Proposed Methodology

The proposed data-driven method is based on the generative adversarial network and the recurrent neural network. In this section, we first elaborate the principles and details of these two networks, and then describe the model architectures and baseline load estimation workflow.

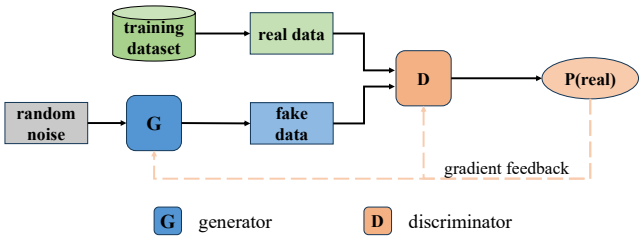

Fig. 2. The structure of GAN.

A. Generative Adversarial Network

Generative models in machine learning algorithms are used to create new samples that have the same statistical properties as the dataset [16]. Since the explicit pattern features of the dataset cannot be extracted, there is no specific error metric for judging the validity of the generated samples. In addition, traditional methods face difficulties in approximating some probability calculations. GAN effectively solves the above problems by a zero-sum game between two sub-models called generator and discriminator, respectively.

The structure of GAN is shown in figure 2. The *generator* is used to generate new data samples that match the training data distribution. It is a model that maps input vector (e.g. noise, incomplete sequences, etc.) to the training dataset space, and is usually implemented by a deep neural network. We denote a prior distribution as $p_{\boldsymbol{z}}(\boldsymbol{z})$ for the input vector $\boldsymbol{z}$, then define the mapping to training dataset space as $G(\boldsymbol{z};\theta_g)$, where G is a differentiable function represented by a deep neural network with parameter θ_g. The objective is to make the generator's distribution p_g over training data as close to the training dataset distribution p_{data} as possible, as follows:

$$\min KL(p_{data}||p_g), \tag{1}$$

where KL is the Kullback–Leibler divergence and has a value of 0 if and only if $p_{data} = p_g$.

The *discriminator* distinguishes whether a sample is real data or generated data. It is similar to a binary classifier that determines the realistic degree of input samples, and is also a deep neural network. We define a model $D(\boldsymbol{x};\theta_d)$ that outputs a single scalar, where θ_d is the model parameter and $D(\boldsymbol{x})$ represents the probability that $\boldsymbol{x}$ is from p_{data} rather than p_g. The objective of the discriminator is to assign correct labels to both the training data and the generated samples.

Based on the above objectives, the general loss functions of the generator and discriminator are formulated with value function $V(G,D)$, respectively, as follows:

$$\min_G V(D,G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))], \tag{2}$$

$$\begin{aligned}\max_D V(D,G) = {} & \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] \\ & + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))].\end{aligned} \tag{3}$$

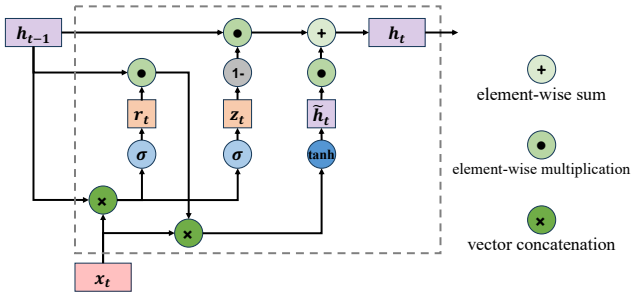

Fig. 3. The architecture of GRU cell.

As $D(x)$ is the probability that the input sample x comes from the real data distribution, the closer $D(G(z))$ is to 1, the more similar p_g is to p_{data}, i.e., it satisfies Eq. (1).

Since the discriminator tries to distinguish the real samples from the generated ones, and the goal of the generator is to fool the discriminator so that it cannot distinguish, a competitive adversarial relationship is formed. We combined Eq. (2) and Eq. (3) to formulate the following two-player minimax game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]. \tag{4}$$

By training the generator and discriminator simultaneously, they are made to play against each other in a zero-sum game, and the training process is iterated until the Nash-equilibrium point is reached. At this point, the discriminator can no longer distinguish between the real data and the samples generated by the generator, which means that the distribution of the two types of data is consistent, i.e., $p_g = p_{data}$.

As mentioned before, $G(\boldsymbol{z};\theta_g)$ and $D(\boldsymbol{x};\theta_d)$ are model parameters of the deep neural network, thus gradient-based learning rules can be exploited to optimize their performance. Further, in order to capture the temporal relationship of consumer baseline load, we choose the recurrent neural network (RNN) as the base model, which is introduced in detail in the following section.

B. Gated Recurrent Unit Network

RNNs are a class of neural networks with short-term memory capability that can process time-series data [18]. Unlike feedforward neural networks in which the output of the network depends only on the current input, the outputs of an RNN not only depend on the current input, but also are influenced by their own historical information, forming a closed-loop structure to realize the memory function. However, due to the gradient explosion or gradient disappearance phenomenon, RNNs can actually only learn the dependencies between short-term states, which is called the long-term dependencies problem. The gated recurrent unit (GRU) network is a variant of RNN, which effectively solves the long-term dependencies problem by introducing the gating mechanism to control the accumulation rate of information [18]. Figure 3 illustrates the architecture of GRU.

In the GRU, the reset gate r_t adjusts the combination of the current input x_t and the previous state h_{t-1} to calculate the current candidate state $\tilde{h}_t$:

$$r_t = \sigma(\boldsymbol{W}_r x_t + \boldsymbol{U}_r h_{t-1} + \boldsymbol{b}_r), \tag{5}$$

$$\tilde{h}_t = tanh(\boldsymbol{W}_h x_t + \boldsymbol{U}_h(r_t \odot h_{t-1}) + \boldsymbol{b}_h), \tag{6}$$

where $\boldsymbol{W}_r$, $\boldsymbol{W}_h$ are the input weight matrices, $\boldsymbol{U}_r$, $\boldsymbol{U}_h$ are the hidden weight matrices, and $\boldsymbol{b}_r$, $\boldsymbol{b}_h$ are the bias vectors. Symbols σ and $tanh$ are nonlinear activation functions, and symbol $\odot$ denotes the element-wise multiplication.

In addition, the update gate z_t is used to control the balance between input and historical information, which is defined as:

$$z_t = \sigma(\boldsymbol{W}_z x_t + \boldsymbol{U}_z h_{t-1} + \boldsymbol{b}_z), \tag{7}$$

where $\boldsymbol{W}_z$, $\boldsymbol{U}_z$ are the input and hidden weight matrices, respectively, and $\boldsymbol{b}_z$ is the bias vector.

Combining Eqs. (5)–(7), the current state h_t of GRU is expressed by:

$$h_t = (1-z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \tag{8}$$

It can be seen that GRU exploits z_t to control the information that the current state needs to retain from h_t and needs to accept from $\tilde{h}_t$. When $z_t = 0$, h_t is only related to x_t and irrelevant with h_{t-1}; When $z_t = 1$, h_t is equal to h_{t-1} and independent of x_t.

Compared with LSTM, GRU has a simpler structure, lower computational complexity and fewer parameters, which can get similar or even better results with fewer training times. Therefore, we adopt GRU as the base network for the generator and discriminator.

C. Model Architecture

Our proposed method is inspired by the GAN and GRU. The generator utilizes the GRU to compress the input incomplete time series x into a low-dimensional vector. Then it uses this vector to generate a complete time series $\boldsymbol{x'}$ hoping to fool the discriminator. While the discriminator adopts the GRU to distinguish between real sequence $\boldsymbol{x}$ and the fake sequence $\boldsymbol{x'}$. They are trained through a min-max game and eventually generate new samples that follow the distribution of the training dataset.

The architecture of the generator in the proposed method is shown in Fig. 4. The generator first consists of a GRU layer to learn the temporal relationship of the input time series $\boldsymbol{x}$. Then a fully connected layer is stacked to the last hidden state of GRU to generate the new complete time series $\boldsymbol{x'}$. In order to distinguish between fake and real samples, the discriminator is also composed of a GRU layer and a fully connected layer. Furthermore, the output of the discriminator is restricted to between 0 and 1 using the sigmoid activation function σ, since the output is a probability indicating the degree of truth.

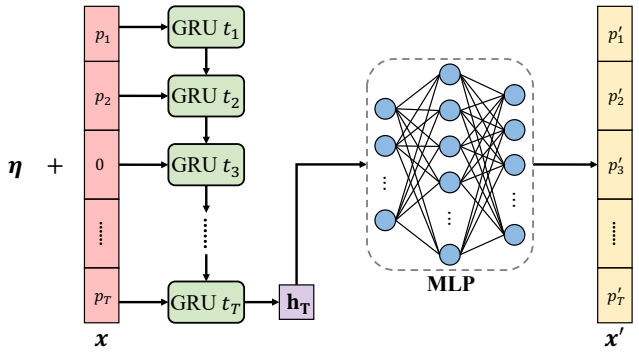

Fig. 4. The illustration of the generator in our proposed method.

D. Baseline Estimation Workflow

1) Data Preparation: We use symbol $\boldsymbol{p}$ to represent the actual load of the consumer for a day, which is expressed as:

$$\boldsymbol{p} = [p_1,\ p_2,\ \ \ldots,\ p_T] \tag{9}$$

where p_t is the actual metered load during time interval t, and T is the number of time intervals in a day (e.g., 24, 48, or 96, depending on the time granularity).

When there is no demand response event, the CBL is equal to the actual load, but when the consumer is involved in DR, its CBL is not equal to its actual load. We consider the data of CBL during the DR periods to be missing and replace the corresponding values with zero, so the metered CBL $\boldsymbol{x}$ can be expressed as:

$$x_t = \begin{cases} p_t, & t \in T_{NDR} \\ 0, & t \in T_{DR} \end{cases}, \tag{10}$$

where T_{DR} and T_{NDR} denote the DR and non-DR periods.

Since the metered CBL $\boldsymbol{x}$ may be incomplete, we introduce a mask vector $\boldsymbol{m}$ to represent whether each value of $\boldsymbol{x}$ exist or not, defined as follows:

$$m_t = \begin{cases} 1, & t \in T_{NDR} \\ 0, & t \in T_{DR} \end{cases}. \tag{11}$$

2) Training Process: Both the input and output of the generator are time series, and we want them to be as close as possible during the non-DR periods. To avoid the generator degenerating into a simple mapping, we add noise to the input $\boldsymbol{x}$ and then obtain the generated data $\boldsymbol{x}'$, which can be formulated as:

$$\boldsymbol{x}' = G(\boldsymbol{x} + \eta), \tag{12}$$

where η is the random noise sampled from a standard distribution, and it has the same size as the input $\boldsymbol{x}$.

Considering that the generator is designed to generate $\boldsymbol{x}'$ similar to $\boldsymbol{x}$, not only their distributions should be the same, but also their specific values should be close. Therefore, the squared error loss is added to the loss function as follows:

$$L_G = \|\boldsymbol{m} \odot \boldsymbol{x} - \boldsymbol{m} \odot \boldsymbol{x}'\|_2 - D(\boldsymbol{x}'), \tag{13}$$

Algorithm 1: The detailed training process of GAN

Input : Parameters of generator and discriminator θ_G, θ_D, batch size B, iteration steps of generator and discriminator k_1, k_2
Output: Updated parameters of generator and discriminator θ'_G, θ'_D

Procedure:
for *total training iterations* **do**
  for k_2 *steps* **do**
    Sample minibatch of B data $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(B)}\}$ from training dataset;
    Generate fake data $\{\boldsymbol{x}'^{(1)}, \ldots, \boldsymbol{x}'^{(B)}\}$;
    Update the parameters of discriminator θ'_D by its gradient $\nabla_{\theta_D} L_D$;
  end
  for k_1 *steps* **do**
    Sample minibatch of B data $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(B)}\}$ from training dataset;
    Calculate the data mask $\{\boldsymbol{m}^{(1)}, \ldots, \boldsymbol{m}^{(B)}\}$;
    Generate the fake data $\{\boldsymbol{x}'^{(1)}, \ldots, \boldsymbol{x}'^{(B)}\}$;
    Update the parameters of generator θ'_G by its gradient $\nabla_{\theta_G} L_G$;
  end
end

where $\|\cdot\|_2$ represents the 2-norm.

In addition, we want the discriminator to output a high probability for $\boldsymbol{x}$ and a low probability for $\boldsymbol{x}'$, so its loss function can be designed as:

$$L_D = D(\boldsymbol{x}') - D(\boldsymbol{x}). \tag{14}$$

Based on Eqs. (13)–(14), we iteratively optimize the generator and discriminator, and finally reach the equilibrium under the adversarial framework. The training details are shown in Algorithm 1.

For an incomplete sequence $\boldsymbol{x}$, we utilize the well-trained generator to generate a complete sequence $\boldsymbol{x}'$, and fill in the missing values of $\boldsymbol{x}$ with the corresponding values of $\boldsymbol{x}'$. The complete CBL $\boldsymbol{x}_{new}$ can be expressed as follows:

$$\boldsymbol{x}_{new} = \boldsymbol{m} \odot \boldsymbol{x} + (1 - \boldsymbol{m}) \odot \boldsymbol{x}'. \tag{15}$$

In summary, we first convert the CBL data to the missing form, then implement the generator and discriminator using GRU and train them by adversarial games. Finally, the CBL estimation is accomplished by merging the generated sequence and the original sequence.

III. Case Studies

A. Experiment Settings

1) Dataset Description: The experiments are conducted on a dataset of buildings in Zhuhai, China. The dataset contains 30 office buildings and hotels, from November 2020 to October 2021 with 30-min granularity.

TABLE I
TRAINING PARAMETERS

Parameter	Description	Value
B	The batch size	16
E	The number of epochs	100
η	The learning rate	0.001
Optimizer	The learning algorithm	Adam
L_G	The loss function of generator	Eq. (13)
L_D	The loss function of discriminator	Eq. (14)

We integrate the 24-hour load of a building (i.e., 48 data points) into a complete CBL sequence. The DR events are simulated by randomly replacing the load with 0 between 14:00 to 19:00. The proportion of DR-event sequences is 20%.

2) Benchmarks and Metrics: To verify the effectiveness of the proposed method, we choose four benchmarks for comparison: 1) averaging method: Low5of10, Mid4of6 and High5of10 [7]; 2) regression method: support vector regression (SVR) [9]. In addition, the mean absolute error (MAE) and mean absolute percentage error (MAPE) are adopted to measure the performance of CBL estimation.

3) Environmental Setup: The proposed method is implemented by an open-source machine learning framework PyTorch[1]. The parameters of the training process are listed in Table I. All the experiments are conducted on a desktop with Intel(R) Core(TM) i7-9700 CPU and NVIDIA GeForce RTX 2080TI GPU (64GB RAM) on a Windows 10 platform.

B. Performance and Comparison of CBL Estimation

In this part, we demonstrate the effectiveness of the proposed method based on the estimation performance on the two types of buildings. Furthermore, the superiority of our proposed method is also validated by comparing it with the abovementioned four benchmarks.

Figure 5 shows the CBL estimation results of our proposed method, where the figures 5(a) and 5(b) are for commercial buildings and the figures 5(c) and 5(d) are for hotels. It can be seen that our method has a high estimation accuracy for commercial buildings, and the estimation CBLs (i.e., red curves) are approximately close to the actual CBLs (i.e., blue curves) at all times except for some fluctuating cases. As for hotels, the performance is relatively poorer due to the higher uncertainty of load profiles. Nevertheless, both the estimated and actual CBLs follow essentially a similar trend. It is worth noting that our method estimates the CBL not only for DR periods but also for non-DR periods. When we adopt Eq. (15) to restore the complete CBL, the recovery performance is further improved since the estimation error in non-DR periods is eliminated. In addition, the well-trained generator will not be disturbed by the zero values during the DR period, although the differences between the zero values (i.e., gray curves) and the corresponding actual CBL (i.e., blue curves) are large. This indicates that our proposed method is robust.

[1]PyTorch: https://pytorch.org/

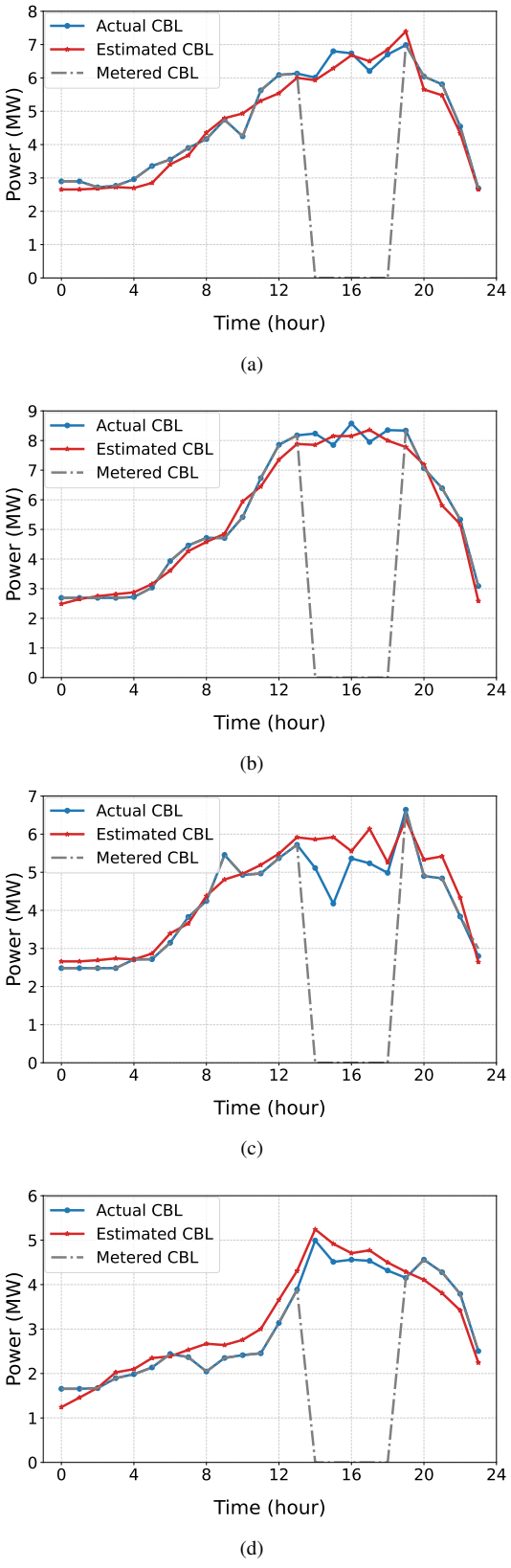

Fig. 5. The CBL estimation results by our proposed method.

TABLE II
PERFORMANCE COMPARISON BETWEEN BENCHMARKS AND THE PROPOSED METHOD.

Method	Commercial Building		Hotel	
	MAE(MW)	MAPE(%)	MAE(MW)	MAPE(%)
Low5of10	0.5617	9.3251	0.6639	13.6831
Mid4of6	0.4601	8.3934	0.5925	11.6320
High5of10	0.5377	9.1890	0.6166	12.7601
SVR	0.3529	6.7035	0.4868	8.9638
Proposed	**0.2703**	**5.4162**	**0.3221**	**7.3563**

We calculate the average performance metrics of our method and the four benchmarks for two types of buildings, shown in Table III-B. The averaging methods, i.e., Low5of10, Mid4of6, and High5of10, directly use average values of historical loads for CBL estimation without considering the volatility and randomness of the load. As a result, their estimation errors are all larger than those of the other two methods. Their smallest MAPE is still over 8%. Compared with the averaging methods, the performance of the SVR method is improved, and the minimum MAE is close to 0.35MW. However, the SVR method still has certain errors because the relationship between load and other factors is difficult to accurately describe. In comparison, our proposed method is apparently superior to the other methods for both commercial buildings and hotels, with a minimum MAPE of less than 5.5% and a maximum MAPE of only around 0.3WM.

IV. CONCLUSION

In this paper, we focus on the CBL estimation problem for DR. In order to overcome the shortcomings of existing methods, we propose a novel CBL estimation method based on GAN. We transform the estimation problem into a time-series data recovery problem, and use the GRU as the basic model to deal with time-series relations. The missing CBLs are restored through the well-trained generator, which is obtained by the adversarial game training. Our method can perform the accurate CBL estimation by filling the generated CBLs into the original incomplete sequence. Case studies indicate that the generated CBL by our method is closer to the real CBL than other benchmarks, which validates its effectiveness and superiority.

REFERENCES

[1] X. Yan, Y. Ozturk, Z. Hu, and Y. Song, "A review on price-driven residential demand response," *Renewable and Sustainable Energy Reviews*, vol. 96, pp. 411–419, 2018.

[2] C. Huang, H. Zhang, Y. Song, L. Wang, T. Ahmad, and X. Luo, "Demand response for industrial micro-grid considering photovoltaic power uncertainty and battery operational cost," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3043–3055, 2021.

[3] D. Ellman and Y. Xiao, "Incentives to manipulate demand response baselines with uncertain event schedules," *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1358–1369, 2020.

[4] F. Wang, K. Li, C. Liu, Z. Mi, M. Shafie-Khah, and J. P. Catalão, "Synchronous pattern matching principle-based residential demand response baseline estimation: Mechanism analysis and approach description," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 6972–6985, 2018.

[5] P. Tao, F. Xu, Z. Dong, C. Zhang, X. Peng, J. Zhao, K. Li, and F. Wang, "Graph convolutional network-based aggregated demand response baseline load estimation," *Energy*, vol. 251, p. 123847, 2022.

[6] K. Li, J. Yan, L. Hu, F. Wang, and N. Zhang, "Two-stage decoupled estimation approach of aggregated baseline load under high penetration of behind-the-meter pv system," *IEEE Transactions on Smart Grid*, vol. 12, no. 6, pp. 4876–4885, 2021.

[7] Y. Zhang, W. Chen, R. Xu, and J. Black, "A cluster-based method for calculating baselines for residential loads," *IEEE Transactions on smart grid*, vol. 7, no. 5, pp. 2368–2377, 2015.

[8] X. Ge, F. Xu, Y. Wang, H. Li, F. Wang, J. Hu, K. Li, X. Lu, and B. Chen, "Spatio-temporal two-dimensions data based customer baseline load estimation approach using lasso regression," *IEEE Transactions on Industry Applications*, 2022.

[9] Y. Chen, P. Xu, Y. Chu, W. Li, Y. Wu, L. Ni, Y. Bao, and K. Wang, "Short-term electrical load forecasting using the support vector regression (svr) model to calculate the demand response baseline for office buildings," *Applied Energy*, vol. 195, pp. 659–670, 2017.

[10] X. Zhou, Y. Gao, W. Yao, and N. Yu, "A robust segmented mixed effect regression model for baseline electricity consumption forecasting," *Journal of Modern Power Systems and Clean Energy*, 2020.

[11] E. Lee, K. Lee, H. Lee, E. Kim, and W. Rhee, "Defining virtual control group to improve customer baseline load calculation of residential demand response," *Applied Energy*, vol. 250, pp. 946–958, 2019.

[12] Y. Deng, L. Wang, H. Jia, X. Tong, and F. Li, "A sequence-to-sequence deep learning architecture based on bidirectional gru for type recognition and time location of combined power quality disturbance," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 8, pp. 4481–4493, 2019.

[13] H. Zang, R. Xu, L. Cheng, T. Ding, L. Liu, Z. Wei, and G. Sun, "Residential load forecasting based on lstm fusing self-attention mechanism with pooling," *Energy*, vol. 229, p. 120682, 2021.

[14] D. Jeong, C. Park, and Y. M. Ko, "Missing data imputation using mixture factor analysis for building electric load data," *Applied Energy*, vol. 304, p. 117655, 2021.

[15] X. Wang, Y. Wang, J. Wang, and D. Shi, "Residential customer baseline load estimation using stacked autoencoder with pseudo-load selection," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 1, pp. 61–70, 2019.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[17] Y. Chen, Y. Wang, D. Kirschen, and B. Zhang, "Model-free renewable scenario generation using generative adversarial networks," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 3265–3275, 2018.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.