Energy data generation with Wasserstein Deep Convolutional Generative Adversarial Networks

Li, Jianbin; Chen, Zhiqiang; Cheng, Long; Liu, Xiufeng

Published in: Energy

Link to article, DOI: 10.1016/j.energy.2022.124694

Publication date: 2022

Document Version Publisher's PDF, also known as Version of record

Link back to DTU Orbit


General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Energy data generation with Wasserstein Deep Convolutional Generative Adversarial Networks

Jianbin Li a, Zhiqiang Chen a, Long Cheng a,*, Xiufeng Liu b

a School of Control and Computer Engineering, North China Electric Power University, 102206, Beijing, China
b Department of Technology, Management and Economics, Technical University of Denmark, 2800, Kgs. Lyngby, Denmark

Abstract

Residential energy consumption data and related sociodemographic information are critical for energy demand management, including providing personalized services, ensuring energy supply, and designing demand response programs. However, it is often difficult to collect sufficient data to build machine learning models, primarily due to cost, technical barriers, and privacy. Synthetic data generation becomes a feasible solution to address data availability issues, while most existing work generates data without considering the balance between usability and privacy. In this paper, we first propose a data generation model based on the Wasserstein Deep Convolutional Generative Adversarial Network (WDCGAN), which is capable of synthesizing fine-grained energy consumption time series and corresponding sociodemographic information. The WDCGAN model can generate realistic data by balancing data usability and privacy level by setting a hyperparameter during training. Next, we take the classification of sociodemographic information as an application example and train four classical classification models with the generated datasets, including CNN, LSTM, SVM, and LightGBM. We evaluate the proposed data generator using Irish data, and the results show that the proposed WDCGAN model can generate realistic load profiles with satisfactory similarity in terms of data distribution, patterns, and performance. The classification results validate the usability of the generated data for real-world machine learning applications with privacy guarantee, e.g., most of the differences in classification accuracy and F1 scores are less than 8% between using real and synthesized data.

© 2022 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

The next-generation smart grid faces challenges for sustainable energy management, which requires bi-directional information flow between customers and energy operators. Smart meters, a type of advanced metering infrastructure (AMI), enable communication between customers and energy operators, which typically record energy consumption at 30-min intervals [1]. Smart meter data analysis helps utilities better understand their customers, enabling them to offer personalized services, design demand response programs, and provision energy supply. Conversely, customers can better understand their energy consumption, improve energy efficiency, and change their consumption behaviors. Among others, smart meter data can be used to energy forecasting [2–4], anomalous consumption detection [5,6], price design [7], demand-side management [8,9] and customer segmentation [10]. Building data-driven models depend on the availability of sufficient data for training. This is especially the case for building deep learning models, which requires a large amount of data for training. However, it is often challenging to obtain sufficient data due to cost, technical barriers and privacy reasons. The release of smart meter data is often strictly controlled and regulated, and data usability is often compromised due to over anonymization. For example, the public data sets [11–15] are anonymized based on the aggregation or removal of the features that can potentially identify individuals. As a result, fine-grained and feature-rich data availability is the main obstacle to building various data-driven models.

In non-intrusive load monitoring (NILM), collecting fine-grained data is cost-prohibitive as it requires installing additional devices to monitor appliance-level consumption for individual households. In addition, real-world energy consumption data can have quality issues, such as missing values and outliers, due to sensors or communication problems in data collection. These missing values and outliers must be corrected before being used to train models. A
The architecture of GAN consists of two networks, a generator and a discriminator. The generator, denoted by $G$, produces synthetic data, while the discriminator, denoted by $D$, estimates the probability that input data is real rather than synthetic. These two networks compete with each other in the training process and eventually find an equilibrium. During the training process, the random noise $Z$ is sent into the generator as the input data, and the generator produces the synthetic data $G(Z)$ as output. In each epoch, the discriminator receives the real sample $X$ or generated sample $G(Z)$ to judge each sample as real or fake, and gives an output $D(X)$ or $D(G(Z))$, respectively. The overall loss of GAN consists of two parts: one from the generator and the other from the discriminator. The generator aims to minimize the following function, while the discriminator aims to maximize it:

$$\mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))]$$

(1)

where $\mathbb{E}_x$ stands for the expected value of all real samples, and $\mathbb{E}_z$ stands for the expected value of all random data to the generator. This formula comes from the cross-entropy between the real data distribution and the generated data distribution. The objective of the discriminator is to maximize the average of the log probability of real samples and the log of the inverse probability of fake samples so that the discriminator can distinguish the difference between real and fake samples:

$$D: \text{maximize } \log D(x) + \log(1 - D(G(z)))$$

(2)

Conversely, the objective of the generator is to minimize the log of the inverse probability predicted by the discriminator for fake data, which encourages the generator to produce samples with a low probability of being fake:

$$G: \text{minimize } \log(1 - D(G(z)))$$

(3)

Throughout the training epoch, the generator and the discriminator take turns involving updates to their model weights according to the descent of the loss function. When network $G$ and network $D$ reach Nash equilibrium, the training process ends. Then, the generator is ready to generate synthetic samples.

Although the structure of the original GAN model is relatively simple, the training of the GAN model is complex, which often suffers from instability, e.g., hard to converge or model collapses. Many attempts have been made to improve the original model. Among others, these include the deep convolutional GAN (DCGAN) [21] which introduces convolutional layers to enhance the feature extraction capability, conditional GAN (CGAN) [22], which adds additional condition vectors to the input of the discriminator and the generator to control the GAN output, the Wasserstein GAN (WGAN) [23], which replaces the original loss function with the Wasserstein distance [24].

Load profile generation is a complex process that requires considering autocorrelation, periodicity, and temporal dependencies of time series. This is very different from image data in terms of complexity and data characteristics. To generate the load profiles, we use an improved GAN model, called WDCGAN, derived from the above three variants: DCGAN [21], CGAN [22] and WGAN [23]. This model employs the Wasserstein distance [24] in its loss function, instead of the common Kullback-Leibler (KL) divergence [25]. The reason is that the Wasserstein distance, as an alternative objective, can properly measure the distance between two datasets no matter whether their distributions are disjoint. It can improve learning stability, eliminate problems such as mode collapse and gradient vanishing, and provide meaningful learning curves for debugging and hyperparameter searches. To better learn the temporal and periodic features of the electricity consumption data, we converted one week’s electricity data into an image-like matrix form and inserted the convolutional layers into the GAN network by referring to the DCGAN model to better extract the temporal and periodic features of the consumption data. To control the type of data generated, we refer to CGAN and add conditional information to the input of native GAN. With the WDCGAN model, users can generate realistic fine-grained scalable load profiles with only small samples as seed (as training set). The generated data can then be used for different purposes, e.g., to train data-driven models, distribution, and sharing without worrying about data privacy. Users can balance the usability and privacy of the data by tuning a hyperparameter during training. As the generated load profiles will be used to train data models or for analysis purposes, the quality is crucial. To evaluate the WDCGAN model, we first study the statistic
characteristics of the synthetic load profiles, including distribution, patterns and auto-relations; and compare with the real data. Then, we use the synthetic data for an application to identify the sociodemographic information of the household based on load profiles, with four classic classification models, including the Convolutional Neural Network (CNN) [26], Long Short-Term Memory (LSTM) [27], Support Vector Machine (SVM) [28] and LightGBM [29]. The experimental results show that the accuracy of these classification models trained with the synthetic data is comparable to those trained with the real data, which confirms the effectiveness of the proposed model.

In summary, the contributions of this paper are threefold:

- Proposes the WDCGAN model for load profile generation. The model addresses data privacy, cost, and technical barrier issues of data management in the energy sector.
- Implements an improved neural network structure in the proposed model, which is capable of improving training stability and combating model collapse issues.
- Showcases the machine learning task of identifying household sociodemographic information based on load profiles, evaluates the model comprehensively using real-world Irish CER data, and validates the effectiveness of the model.

The remainder of this paper is organized as follows: Section 2 reviews the related work. Section 3 introduces the Irish CER data, the sociodemographic identification problem of the household, and presents the details of the model, the identification task, and evaluation metrics. Section 4 conducts experiments to evaluate the model. Section 5 concludes the paper and presents future work.

2. Related work

This section will first review the load profile generation methods; then discuss the application of the generated data sets, which in this case is for the identification of household sociodemographic information.

### 2.1. Load profile generation

Load profile generation has received a lot of attention in the past. Table 1 lists some methods for generating load profiles, but not exhaustively. These methods can be classified into two broad categories: mathematical modeling methods and data-driven methods. Most of the earlier works are based on simulation using mathematical modeling. The main benefit is that the simulation methods do not require real samples to generate data. However, simulation methods require complex mathematical modeling knowledge and the generated load data are generally less accurate. The most widely used mathematical method is the Markov model. Therefore, to better improve performance, external sources become necessary to assist the models, such as appliances [32], physical characteristics of buildings [31], human activities [30–32], or weather conditions [33]. In recent years, data-driven methods become increasingly popular, mainly due to better accuracy and the availability of some public data sets. Data-driven methods typically require some real samples as seed or training the model for data generation. A more classic data-driven method is regression models trained with sample data, e.g., Refs. [34,35]. With emerging deep learning, neural network-based methods appear, such as LSTM-based [37], ANN-based [41] and GAN-based methods [38–40]. In this paper, we favor the GAN-based method optimized with the Wasserstein distance in its loss function, because of its excellent performance in simulating realistic data. Data-driven approaches can generate scalable data with a given relatively small-sized real-world data as the seed for training. In addition, there are some other methods used for load profile generation, e.g., the graph signal processing (GSP) based method, e.g., Ref. [43] and the simple arithmetic method, e.g., Ref. [36]. But these methods only derive new load profiles based on the given samples, which are not suitable for generating scalable data sets like the prediction-based methods, such as regression or neural networks.

The methods can be divided into top-down and bottom-up approaches in terms of how load profiles should be generated. Top-down approaches typically generate fine-grained load profiles based on disaggregation, e.g., by dividing the total consumption of a

<table>
<thead>
<tr>
<th>Ref. Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[30] Markov chain-based</td>
<td>Using a Markov chain-based approach to simulate user behavior for home energy consumption modeling to generate data;</td>
</tr>
<tr>
<td>[31] NILM-based and Markov Chain</td>
<td>Use a simulation model to generate load profiles by incorporating physical characteristics and behavioral factors; A bottom-up approach;</td>
</tr>
<tr>
<td>[32] Simulation model</td>
<td>Use a simulation model to generate load profiles, what can match appliance use and indoor activities; Synthesize down to 1-min level load profiles.</td>
</tr>
<tr>
<td>[33] Discrete-event simulation combined with a stochastic method</td>
<td>Generate load profiles for most energy-intensive subsectors, such as a steel factory; Model the timely behaviours of production processes; Generate load profiles with varying time-resolution from minute to daily.</td>
</tr>
<tr>
<td>[34] Regression-based &amp; Markov simulation model</td>
<td>Synthesize hourly consumption data with small samples as the seed.</td>
</tr>
<tr>
<td>[35] Regression-based</td>
<td>Use a predictive model to generate load profiles based on energy consumption patterns;</td>
</tr>
<tr>
<td>[36] Arithmetic sum</td>
<td>Generate aggregated plug load profiles in office buildings; Generate building-level aggregated plug load profiles for different building types; Input use-defined parameters including occupants, activities, location and appliance type.</td>
</tr>
<tr>
<td>[37] Mixture density network (MDN) &amp; LSTM</td>
<td>Integrate MDN with LSTM to generate hourly load profiles.</td>
</tr>
<tr>
<td>[38] Recurrent GAN</td>
<td>Use RGAN to generate load profiles with real samples as the training set; Test on appliance and building level data; Evaluate the model on prediction application.</td>
</tr>
<tr>
<td>[39] GAN-based</td>
<td>Proposing a GAN-based scheme to generate synthetic labeled load patterns and usage habits</td>
</tr>
<tr>
<td>[40] GAN-based</td>
<td>A model-free method for scenario generation of smart grid and GAN is used to capture the spatial and temporal correlations of renewable</td>
</tr>
<tr>
<td>[41] ANN-based</td>
<td>Synthesize load profiles based on publicly available load and weather data; Use the prediction method based on ANN</td>
</tr>
<tr>
<td>[42] Demand models of appliances</td>
<td>Use a three-layered approach to disaggregate consumption from urban feeders to appliance; Generate down to 1-min level fine-grained domestic load model per consumer.</td>
</tr>
<tr>
<td>[43] Graph signal processing (GSP)-based</td>
<td>Disaggregate total domestic energy consumption down to individual appliances using GSP; Use a novel event-based graph approach;</td>
</tr>
</tbody>
</table>
load profile and dividing the total consumption into individual households [42] or appliances [43].

Top-down approaches require historical data for training and supplementary data (e.g., weather and indoor activities) for the calibration of load profiles; otherwise, accuracy is poor [34]. Bottom-up approaches, on the other hand, aggregate detailed load profiles at higher levels, e.g., from appliances to an individual household [44,45]. Although bottom-up approaches can generate realistic load profiles with diversity, they have strict requirements for detailed household information. Therefore, both the top-down and bottom-up approaches have limitations in accuracy or complexity.

2.2. Studies on load profiles and sociodemographic information

Sociodemographic information is a type of customer characteristics related to household factors, such as employment status, income, family size, family composition, and appliances. There are two main research directions between sociodemographic information and load profiles: one is to estimate load profiles based on sociodemographic information; and the other is to infer sociodemographic information based on load profiles.

The studies [46–48] found that sociodemographic information has a significant impact on their energy consumption. Huebner et al. [49] confirmed an increasing number of occupants can lead to an increase in total electricity consumption. McLoughlin et al. [50] analyzed customer age as a factor in electricity consumption and mentioned that family members under the age of 36 consumed the most in the evening. Beckel et al. [51] investigated how household composition affects consumption patterns and the total amount. The studies [52–55] disaggregated residential electricity consumption based on sociodemographic information such as appliances and indoor activities.

The studies [51,56,57] suggested that sociodemographic characteristics can be identified from energy consumption data. The sociodemographic factors, including home occupancy, number of people, and number of appliances, have been successfully inferred. For example, several recent studies [58–64] revealed sociodemographic information from smart meter data, including the number of residents, income and occupancy status.

As a result, the study of energy consumption and sociodemographic data is crucial for demand-side energy management, which involves data privacy issues. Data generation can be a viable solution to preserve privacy, which this paper aims to achieve.

3. Methodology

This section will first describe data collection and its processing, then present an overview of the privacy-preserving framework for load profile generation, followed by the detailed description of the proposed WDCGAN model, and finally present the model application and evaluation metrics.

3.1. Data collection and processing

This study uses the Irish CER dataset [14], gathered as a part of the Smart Metering Project initiated in 2007. The purpose of the CER was to investigate the performance of smart meters and their impact on customer energy consumption. This dataset consists of individual electricity consumption data and survey data. The consumption data were recorded by smart meters at a 30-min interval from July 2009 to December 2010 (75 weeks). Fig. 2 shows the load profiles of three representative households over a week. The survey data contains the sociodemographic information collected from 4223 households by filling out the multiple-choice questionnaire. This sociodemographic information includes occupancy status (e.g., employment status, social class), consumption behavior (e.g., interest in reducing bill), household properties (e.g., floor area, number of bedrooms) and home appliances (e.g., number of washing machines). This paper selects ten representative sociodemographic characteristics from the 140 questions in the survey data as shown in Table 2. We have removed the rows with missing data or specific classes involving attributes that do not represent a significant proportion of customers. The customers’ answers to these questions were used as class labels in our identification task and generation work. The corresponding question numbers in the survey data are provided in the second column of the table. The last two columns provide the classification rules for each question and the number of customers in each category. Since data generation and social demographic identification require a large amount of training data, we segment consumption time series by week to enrich the training dataset.

3.2. Framework overview

Fig. 3 shows the WDCGAN-based privacy-preserving framework for sociodemographic information identification. This framework can be divided into two parts: data generation and sociodemographic information identification. The generated data can be used to replace the real-world data for classification tasks in machine learning. The similarity between the real data and the generated data will be calculated in order to evaluate the performance of the data generator indirectly. This figure shows the nine critical blocks and their interaction in the framework, which are described in the following.

- **Block 1** is the input for training or testing, including electricity consumption data and the corresponding sociodemographic data.
- **Block 2** is the generative model to be trained. The generative model used is WDCGAN and the training data are from block 1.
The generated data have the same distribution as the real data in block1, and serve as the training data for the classification model in block 4. Block 4 is the classification model to be trained by the generated data from block 3. The classification models include CNN, LSTM, SVM and lightGBM.

Block 5 is the trained classification model from block 4, which receives the real data from block6 and makes the prediction of the sociodemographic label.

Block 6 is the processed real electricity consumption data, which will be sent to block 5 as input of the trained classification model.

Block 7 is the real customer sociodemographic information used to validate the classification model.

Block 8 is applied to compare the predicted labels from block 5 with the actual labels from block 7. Block 9 is the comparison result of the prediction accuracy of all classification models in real and synthetic data.

The nine blocks can be divided into two parts. The first part includes four blocks for data generation (block1 – block3) and classification training (block4). The second part includes the blocks from 5 to 9 for the evaluation. There are two types of models that must be trained in our framework. One is the generative model for generating data, while the other is the classification model for evaluating the model. The classification model will be trained using synthetic data from the generative model. In the evaluation process, the resulting classifier is used to predict the labels with the given real consumption data as input (block6), and the prediction labels, i.e., \( \hat{y} \), will be compared against the real labels, i.e., \( y \) (block8)
to evaluate the model performance (block9). In general, the higher the accuracy, the more usable the data, but with less privacy. Therefore, it is essential to balance the usability and privacy of the data. This can be achieved by adjusting a hyperparameter to train the WDCGAN model, and different data generators can be trained to generate data to meet different user requirements.

3.3. Proposed WDCGAN model

As discussed earlier, GANs can learn the distribution of a given dataset, and have been used successfully in graphical and natural language processing. However, the application of the original GANs to load profile generation for identifying sociodemographic information still faces some challenges. First, the original GANs cannot control the generated data after training because the generator input to the GANs is only noise, which means that the generated data lack label information. For the identification of sociodemographic information, if the generated data lacks label information, the mapping between the generated consumption data and consumer's labels cannot be established, so the generated data cannot be used by classification models. In addition, the greater freedom of the generative model may lead to a larger difference between the generated samples and the real samples, resulting in lower usability of the data and poor stability during the learning process. Second, it is challenging to train GANs because the evolution of discriminator and generator networks must be balanced during the training process. If one of them is much stronger than the other, it will lead to two types of failures: model collapse and convergence failure. That is, if the discriminator is too strong, the generator can only create one type of output or a small set of outputs, resulting in model collapse. Conversely, if the discriminator is too weak, the generator will not be able to produce good quality outputs, resulting in convergence failure. Third, the learning process for generative adversarial networks is not sufficiently constrained. The discriminator only criticizes the probability of true and false for the generated data, but lacks the ability to control the generated data after training because the generator input to the discriminator is a value of 1, which is a judgment of the authenticity and matching of the combined vector. The discriminator function is defined as below:

$$\min_{G} \max_{D} V(D, G) = E_{x \sim p_{data}(x)} [\log D(x|C)] $$
$$+ E_{z \sim p_{z}(z)} [\log (1 - D(G(z|C)))]$$ (4)

where $x$ represents the real data, $z$ represents the random noise data, $D(x|C)$ is the probability that the discriminator judges the real sample $x$ is true under the condition of the given $C$, $G(z|C)$ is the probability when the discriminator judges the generated sample $z$ is true under the condition of the given $C$.

Fig. 4 illustrates the structure of the discriminator and the processing flow. First, the label (i.e., the sociodemographic information) and the consumption data of one week with a granularity of 30 min are all converted into a $7 \times 48$ matrix. The two matrices are connected as input for the discriminator. Then, the four convolutional layers of the discriminator will extract features from the input. The stride size and the kernel size of the convolution layer are defined by the experiments shown in Table 3. Last, the output of the discriminator is a value of $1 \times 1$, which is a judgment of the input data. The discriminator loss is used to guide updates to the discriminator and generator parameters. Only when the generated sample combined with its label is sufficiently similar to the actual sample combined with its label will the discriminator output be true because it has been confused.

Generator: The input of the generator in WDCGAN is a vector composed of random noise and the label with the embedded format. The generator output is a matrix of $7 \times 48$, corresponding to a week of consumption data with a granularity of 30 min. The structure of the generator and the data processing flow are shown in Fig. 5. First, the labels (i.e., sociodemographic) are transformed into $100 \times 1$ by one-shot coding and then concatenated with $100 \times 1$ of random noise data as the generator input. Then, four layers of the transposed convolution are used to perform bottom-up sampling from the input data to the hidden features. The stride size and kernel size of the transposed convolution layers can be found in Table 3. After the four transposed convolution layers, the input vector of $200 \times 1$ is transformed into a matrix of $7 \times 48$. Finally, the $7 \times 48$ matrix is sent to the discriminator as input data during model training or used to generate the required data after training.

With the above carefully designed WDCGAN, we can generate a large amount of synthetic data by controlling the input labels as a hyperparameter for the generator. The input labels and the generated load profiles form matching pairs, replacing the real sociodemographic data and their consumption data. This means that WDCGAN can generate the data used for various supervised
Table 3
The architecture and hyper-parameters of discriminator and generator.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Layer Type</th>
<th>Hyper-parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input Size</td>
<td>Kernels Number</td>
</tr>
<tr>
<td><strong>Discriminator</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>Convolution</td>
<td>$2 \times 7 \times 48$</td>
</tr>
<tr>
<td>D2</td>
<td>Convolution</td>
<td>$16 \times 4 \times 25$</td>
</tr>
<tr>
<td>D3</td>
<td>Convolution</td>
<td>$32 \times 2 \times 12$</td>
</tr>
<tr>
<td>D4</td>
<td>Convolution</td>
<td>$4 \times 1 \times 6$</td>
</tr>
<tr>
<td><strong>Generator</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G1</td>
<td>Transposed Convolution</td>
<td>$200 \times 1$</td>
</tr>
<tr>
<td>G2</td>
<td>Transposed Convolution</td>
<td>$256 \times 1 \times 6$</td>
</tr>
<tr>
<td>G3</td>
<td>Transposed Convolution</td>
<td>$128 \times 2 \times 12$</td>
</tr>
<tr>
<td>G4</td>
<td>Transposed Convolution</td>
<td>$128 \times 2 \times 12$</td>
</tr>
</tbody>
</table>

Fig. 4. The discriminator architecture in the proposed WDCGAN.

Fig. 5. The generator architecture in the proposed WDCGAN.
learning applications with different privacy requirements by adjusting the label hyperparameter.

The WDCGAN process is divided into data preprocessing, model training, and data generation. Data preprocessing involves cleaning, normalizing, and transforming data. Model training involves alternating weight updates between discriminators and generators, where the Wasserstein distance [23] is used to speed up training convergence and stabilize the model. The data generation uses finely trained WDCGAN to generate different categories of consumption data based on the input data.

For data preprocessing, we proceed as follows: First, we obtain the labels from the survey data containing the customer sociodemographic information. Then, we partition energy consumption time series by week and the corresponding label of each customer. The results in 75 weeks as the CER data are at 30-min intervals, which have 336 values for one week. Third, we drop the weeks with the values of continuous null or 0. Fourth, we normalize the data within [0,1] to better train the WDCGAN. Finally, we convert the 336 data values of a week into a $7 \times 48$ matrix. The example of the resulting time series and corresponding heatmap can be found in Figure B.10 in Appendix B.

For the training of WDCGAN, the process is different from that of a general deep learning network, as it has two different networks, a discriminator and a generator, whose parameters must be updated in turn. Since we use WGANs [65], the slow convergence problem of deep learning networks can be overcome. The loss function of WDCGAN can be viewed as the Wasserstein distance with a gradient penalty, which is defined as follows:

$$L = \mathbb{E}_{z \sim \mathcal{P}_z} [D(z)] - \mathbb{E}_{x \sim \mathcal{P}_r} [D(x)] + \lambda \mathbb{E}_{z \sim \mathcal{P}_z} \left(\|\nabla_2 D(z)\|_2 - 1\right)^2$$

Equation (5) is the Wasserstein distance between the generator output $z$ and the real samples $x$ probability distributions $\mathcal{P}_g$ and $\mathcal{P}_r$. This WDCGAN loss function must be minimized during learning. The gradient penalty is a penalty on the gradient norm on the samples where $\mathcal{P}_z$ is the distribution obtained by uniform sampling along a straight line between the real and generated distributions $\mathcal{P}_g$ and $\mathcal{P}_r$. In this paper, we use the following hyperparameters: the gradient penalty coefficient $\lambda = 10$, the number of critic iterations per generator $n_{\text{critic}} = 5$, and the learning rate $\alpha = 0.0001$. For data generation, we use random data of $100 \times 1$ and $n$ labels to feed a well-trained generator and obtain a total of $n$ of $7 \times 48$ output data. Then, we denormalize the output data and obtain the generated load profiles.

3.4. Application of the proposed model

The sociodemographic information identification of customers is essential to build a classification model. This model can find the
mapping between the load profiles (recorded by smart meters) shown in Fig. 2 and the corresponding sociodemographic characteristics shown in Table 2. The consumption time series are denoted by \( X = [X_1, X_2, \ldots X_i, \ldots X_n] \) as input vector, and sociodemographic characteristics are denoted by \( Y = [Y_1, Y_2, \ldots Y_i, \ldots Y_n] \) as labels. The \( n \) and \( i \) indicate the total number of customers and the \( i \)-th customer. \( Y_j \) stands for the \( j \)-th characteristics of all sociodemographic characteristics, and \( Y_j \) means the \( j \)-th characteristics of the \( i \)-th customer. Mathematically, a classification model, \( F_j \), is trained based on the consumption data \( X \) and the label \( Y_j \) for the \( j \)-th characteristics:

\[
Y_j = F_j(w_j, X_i)
\]

where \( w_j \) and \( Y_j \) are the parameters to be learned and the given label of the \( j \)-th characteristics, respectively.

In most previous studies, supervised learning models (e.g., classification and regression models) are trained successfully with the assumption that both \( X \) and \( Y \) require a relatively large dataset, and can be easily obtained. However, as mentioned earlier, access to energy consumption data can be restricted due to privacy concerns.

Recently, GANs have shown excellent performance in producing realistic data with the given training data. Thus, the generated data can be freely shared, accessed, or even used to augment or enrich similar datasets without concern for privacy.

Identifying sociodemographic information is a classification problem, i.e., finding the match from load profiles to sociodemographic information. This study often helps in energy demand-side management. For example, a better understanding of sociodemographic information allows one to provide personalized services and improve energy efficiency, which can benefit both customers and utilities. Therefore, this section will take the identification of sociodemographic information as an example application to evaluate the proposed model. We employ the following four classical classification methods in this application, including CNN, LSTM, SVM, and LightGBM, which are described below. We will train these classification models using real-world consumption data, as well as generated consumption data, and make a comparison to evaluate the proposed WDGCAN.

**CNN:** CNN is a feedforward deep neural network consisting of an input layer, an output layer and several hidden layers (its network structure can be found in Figure B.11 in Appendix B). The hidden layers are typically composed of a convolutional layer, a pooling layer, and a fully connected layer, which are used to process the features obtained from the previous layer. The convolutional layer is locally connected and has shared weight properties, making the CNN invariant to change and space. Compared to other feedforward neural networks, CNN has fewer parameters, so it is faster to train using backpropagation algorithms. CNN uses relatively less pre-processing and can achieve exceptional performance in these tasks. It has become the dominant model in computer vision. In this paper, we define a CNN as a classification model for identifying sociodemographic information. The input layer of the model receives an image in the format \( 7 \times 24 \), which is generated by WDGCAN. According to Ref. [64], the output layer of the model uses an SVM to predict the class based on the features of the hidden layers. In our model, the CNN has two convolutional layers, a max-pooling layer, and two fully connected layers between the input layer and the output layer. The kernel size of the two convolutional layers is \( 3 \times 2 \) and \( 3 \times 3 \), respectively, and the stride is 1. The kernel size of the max-pooling layer is \( 2 \times 2 \), and the stride is 2. The number of cells of the two fully connected layers is 320 and 32, respectively.

**LSTM:** LSTM has a particular recurrent neural network (RNN) that is robust to learning and predicting sequential data. The network can determine what to remember, forget, and generate output by adding gate memory units. LSTM can overcome the problem of RNNs, which are limited in maintaining long-term memory over time. LSTM has good performance in many applications with inherent sequences. LSTM network structure consists of an LSTM layer, a fully connected layer and a softmax layer (see Figure B.12 in Appendix B). The softmax layer is also a fully connected layer, used for classification by mapping the network output to \( (0,1) \). The LSTM layer has 24-dimensional input features and 100 hidden units in this model. The output of the network is the predicted value of the input data class.

**SVM:** SVM is a supervised learning method for classification. SVM uses a kernel function to map the space of input samples to a high-dimensional feature space and then finds the optimal classification surface in the high-dimensional space to obtain a nonlinear relationship between the input and output data. It has a comprehensive theoretical basis and fundamental framework, which can address machine learning problems such as high-dimensionality and overlearning. This paper uses an SVM with a radial basis function (RBF) kernel.

**LightGBM:** LightGBM is, by default, a gradient-optimized decision tree algorithm used for classification. It applies a highly optimized histogram-based decision tree algorithm and offers significant advantages in efficiency and memory consumption. A LightGBM algorithm uses two novel techniques, called gradient-based one-sided sampling (GOSS) and exclusive feature bundling (EFB), which enable the algorithm to run faster with high accuracy. The algorithm can typically significantly speed up the training process, resulting in a more efficient model in many cases. The implementation of these classification models consists of two parts, model training and model evaluation. We use the generated load profiles to train the models, but use the real load profiles to test the trained models. To reduce the model parameters, we aggregate the generated data into hourly data, which changes the input format from \( 7 \times 48 \) to \( 7 \times 24 \). As a result, all real data used for the evaluation are transformed into a \( 7 \times 24 \) matrix.

The input to CNN is one week of consumption time series, \( 7 \times 24 \), and the output is predicted sociodemographic information. The loss function of the CNN is the cross-entropy loss between the predicted label and the generated label. The stochastic gradient descent method is used for parameter optimization during model training and the learning rate is set to \( \alpha_{CNN} = 0.02 \). LSTM will use the same input as the CNN, and uses cross-entropy as the loss function. The LSTM model is trained with Adam for parameter optimization, and the learning rate is set to \( \alpha_{LSTM} = 0.01 \). As for the other two classification models, SVM [66] and LightGBM [29], we will not present their structures and parameters here because they are well-known and widely used models.

### 3.5. Evaluation metrics

To evaluate the performance of the proposed model, we use the formula metrics and the effects of the actual application to evaluate the generated load profiles. The metrics include the autocorrelation of the generated data and the root mean square error (RSME) of the generated data compared to the real data. The actual application effect is the classification accuracy using the generated data.

**Autocorrelation:** Ideally, the generated data should have the same randomness, volatility, and periodicity as the real data. Thus, we evaluated the model by calculating the autocorrelation values of the generated load profiles. The peak value and the valley value represent the periodicity and randomness of the generated data in the autocorrelation curve. A generated time series is represented by \( \{x_t\}_{t=1}^N \), then its autocorrelation value with a lag of \( h \) can be
calculated as follows:

\[
\text{Autocorr}(h) = \frac{1}{N-h} \sum_{t=1}^{N-h} (X_t - \mu)(X_{t+h} - \mu) \quad (7)
\]

where \(\mu\) and \(\sigma\) represent the mean and standard variation, respectively. In addition, the generated data should be diverse and reflect all possible customer electricity consumption scenarios. The scatterplot of each generated dataset’s mean and standard variance can reflect its diversity. The more scattered the scatterplot, the more diverse the generated load profiles are.

**RSME:** The generated data should have characteristics similar to those of the real data. To check the similarity between the generated and the real data, we computed the root mean square error (RSME) between the real time series \(\{X_t\}_{t=1}^{N}\) and the generated time series \(\{X_g\}_{t=1}^{N}\), defined as:

\[
\text{RMSE}(X_t, X_g) = \sqrt{\frac{1}{N} \sum_{t=1}^{N} (X_t^r - X_g^r)^2} \quad (8)
\]

The purpose of the data generated is to be used by various applications. Therefore, we use the accuracy of the generated data in different classification models as an evaluation criterion. Therefore, the generated data are used as the training set for the classification model, and the real data are used as the test set. The difference in classification accuracy in the two data sets may reflect the similarity between real and generated data. The confusion matrix calculates the accuracy of the classification model on the two datasets. For a binary classification task, for example, its confusion matrix is as follows:

\[
\begin{bmatrix}
TP & FN \\
FP & TN
\end{bmatrix}
\quad (9)
\]

where TP (true positive) and TN (true negative) represent the number of positive and negative instances correctly classified by the model. FN (false negative) and FP (false positive) represent the number of incorrectly classified positive and negative instances, respectively. Accuracy is used to evaluate the prediction quality of the classification model, calculated as the proportion of all correctly predicted samples, i.e., TP and TN, to the total number of samples, i.e.:

\[
\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \quad (10)
\]

An accuracy value close to 1 means a good classification result, while an accuracy value close to 0 means a bad classification result. However, the accuracy value of the classification model cannot directly reflect the quality of the generated data. We, therefore, use \(\Delta\text{ACC}\) to evaluate the quality of the generated data, which represents the distance of the accuracy of the same classification model between the real data and the generated data, defined as follows:

\[
\Delta\text{ACC} = \text{Accuracy(Generated)} - \text{Accuracy(Real)} \quad (11)
\]

where \(\text{Accuracy(Generated)}\) represents the accuracy value of the classification model on the generated data, while \(\text{Accuracy(Real)}\) represents the accuracy value of the same classification model on the real data. If the value of \(\Delta\text{ACC}\) is close to 0, it means the performance of using the generated data has no difference to that of using the real data.

**F-measure:** It is also called \(F_1\)-score or balanced \(F\)-score, which stands for the harmonic mean of precision and recall [67,68] as in Eq. 12. It is often used to evaluate classification performance on imbalanced labeled data. The value of \(F_1\)-score ranges from 0 to 1, where a higher value reflects better classification performance.

\[
F_1 = \frac{2 \times \text{Pr} \times \text{Re}}{\text{Pr} + \text{Re}} \quad (12)
\]

where Pr and Re denote precision and recall, respectively, which are calculated as follows:

\[
\text{Pr} = \frac{TP}{TP + FP} \quad \text{Re} = \frac{TP}{TP + FN} \quad (13)
\]

Since the single \(F_1\)-score value on the generated data cannot adequately reflect the quality of the generated data, we define \(\Delta F_1\) as the metric to evaluate the quality of the generated data. It represents the distances of the \(F_1\)-scores of the same classification model in the real and synthetic datasets, reflecting the performance difference on the two datasets. \(\Delta F_1\) is calculated as:

\[
\Delta F_1 = F_1(\text{Generated}) - F_1(\text{Real}) \quad (14)
\]

where \(F_1(\text{Generated})\) represents the \(F_1\) value of the classification model on the generated data, while \(F_1(\text{Real})\) represents the \(F_1\) value of the same classification model on the real data. If the value of \(\Delta F_1\) is close to 0, it means that the performance of using the generated imbalanced data does not differ from that of using the real imbalanced data.

4. Results and analysis

This section will first present the data preparation procedure and experimental settings, then the description of data preparation, and finally evaluate the model based on load profile statistics and sociodemographic information classification using the generated datasets.

4.1. Data preparation and experimental settings

The experiments use the CER data set described in Section 3.1, it records electricity consumption data and sociodemographic information from customers for more than 5000 Irish households from July 14, 2009, to December 31, 2010. After data cleaning, more than 300,000 CER records can be used for the experiment and \(S_q\) represents these cleaned data. Each sample in \(S_q\) contains one week’s time series consumption values and the corresponding sociodemographic information. For the generative model, i.e., WDCGAN, 10,000 or 40,000 samples are randomly selected from \(S_q\) as its training set. Here, we use \(G_0(10000)\) and \(G_0(40000)\) to denote the generators trained with 10,000 and 40,000 real samples, respectively, where \(Q_i\) stands for ten Questions in Table 2. Take \(G_{Q_3}(10000)\) and \(G_{Q_3}(40000)\) as examples, they represent the generators in WDCGAN trained for Question #310 with 10,000 or 40,000 samples from \(S_{R_i}\), respectively. For classification models, 40,000 real samples are randomly selected from \(S_{R_i}\) as their test set (or training set) for the four types of classification models, denoted by \(S_{R_i}Q_i\). At the same time, the trained WDCGAN will use its generator to produce 40,000 synthetic samples, denoted by \(S_{R_i}Q_i(10000)\) or \(S_{R_i}Q_i(40000)\), as training set (or test set) for the four different types of classification models. generated.

4.2. Load profile evaluation based on statistics

As mentioned above, the generator in the trained WDCGAN can be used to produce synthetic data. We can control the conditional vector fed into the generator to produce various synthetic data, satisfying data privacy and data usability requirements.
S\textsubscript{R, Qi}(10000) and S\textsubscript{R, Qi}(40000) represent the 40,000 synthetic data generated by the data generators G\textsubscript{Qi}(10000) and G\textsubscript{Qi}(40000), respectively. The generated datasets S\textsubscript{R, Qi}(10000) and S\textsubscript{R, Qi}(40000) for each question have an identical proportion of sociodemographic categories as the S\textsubscript{R, Qi}, achieved by controlling the number and proportions of categories in the conditional vectors. Take the question #310 as an example. It has two label categories, class-1 and class-2. Therefore, there are a total of 12,488 class-1 samples and 27,580 class-2 samples among the 40,000 selected samples. To reduce overfitting of the classification model, the original time series with a fine granularity of 30 min are aggregated into hourly resolution, so that each customer’s weekly consumption data become a 168-dimensional vector.

Table 4 presents the RSME results between the real data and the data generated for each question. Column2 and Column3 indicate the RSME of S\textsubscript{R, Qi}(10000) and S\textsubscript{R, Qi}(40000) with S\textsubscript{R, Qi}, respectively, and the Column4 is the RSME difference between Column2 and Column3, generated.

For an intuitive comparison between the generated and real data, we plot the mean and standard deviation curves for question #310 in Fig. 6. As shown in Table 2, question #310 has two different categories with an answer, yes or no, and we use label #1 and label #2 to denote yes and no, respectively. It can be seen that the curves of the generated data and the actual load profiles for each class are close to each other and have the same periodicity. However, the mean and standard deviation of the generated data are more volatile than the real data. It is because WDCGAN tries to randomize the generated data, which can be confirmed by the scatterplot in Figs. 7 and 8. The generated data and the real data have similar areas of scattering distribution, suggesting that the generator can successfully generate highly similar consumption data. However, it can also be seen from the two figures that the scattered areas of the real data surround the scattered areas of the generated data, which means that the generated data are less diverse than the real data. A comparison of the two figures shows that the overlap area in Fig. 7 is larger than the overlap area in Fig. 8, which indicates that more training data can help WDCGAN better understand and learn the diversity of actual data. Fig. 9 compares the autocorrelation between the generated data and the real data. Both autocorrelation curves have similar valleys and peaks at adjacent points, which implies that they have the same randomness over the period of a day (lag = 24), confirming that our WDCGAN has learned the periodic characteristics from the real data.

### 4.3. Load profile evaluation based on applications

Table 5 presents the accuracy performance of CNN and SVM in

<table>
<thead>
<tr>
<th>Question No.</th>
<th>Class</th>
<th>Accuracy (Real as Train Real as Test)</th>
<th>F1-Score (Real as Train Real as Test)</th>
<th>ΔF1 – Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CNN</td>
<td>SVM</td>
<td>CNN</td>
</tr>
<tr>
<td>300</td>
<td>2</td>
<td>70.4</td>
<td>67.8</td>
<td>67.2</td>
</tr>
<tr>
<td>310</td>
<td>2</td>
<td>76.6</td>
<td>69.3</td>
<td>69.1</td>
</tr>
<tr>
<td>401</td>
<td>3</td>
<td>58.1</td>
<td>52.5</td>
<td>44.4</td>
</tr>
<tr>
<td>410</td>
<td>2</td>
<td>78.4</td>
<td>81.2</td>
<td>84.3</td>
</tr>
<tr>
<td>450</td>
<td>2</td>
<td>62.2</td>
<td>58.6</td>
<td>52.4</td>
</tr>
<tr>
<td>453</td>
<td>2</td>
<td>66.3</td>
<td>57.8</td>
<td>57.6</td>
</tr>
<tr>
<td>460</td>
<td>4</td>
<td>52.2</td>
<td>48.3</td>
<td>44.1</td>
</tr>
<tr>
<td>4704</td>
<td>2</td>
<td>70.7</td>
<td>69.8</td>
<td>69.9</td>
</tr>
<tr>
<td>4905</td>
<td>2</td>
<td>57.2</td>
<td>52.9</td>
<td>51.1</td>
</tr>
<tr>
<td>6103</td>
<td>3</td>
<td>63.6</td>
<td>63.1</td>
<td>61.7</td>
</tr>
</tbody>
</table>

Fig. 7. Mean and standard variation of the real and generated load profiles for the question #310 with G\textsubscript{Qi}(10000).
different sociodemographic identification tasks. Column3 and Column4 in Table 5 are the accuracy when the model is trained and tested by real data from \( S_{R,Q_i} \); while Column5 and Column6 are the accuracy when the model is trained by the generated data, \( S_{G,Q_i}^{10000} \), but tested by real data \( S_{R,Q_i} \). The last two columns, \( \Delta \text{ACC} \), are the differences between Column3, Column4 and Column5, Column6. From \( \Delta \text{ACC} \), we can find that the differences in accuracy between the real and generated datasets for the 10 classification tasks are mostly less than 8.0%, only the questions, #401, #450 and #453, with a greater difference.

As the labels for the 10 classification tasks are imbalanced, \( F_1 \) is used for a better assessment of the imbalanced data. The results are shown in Table 6. The experimental settings of Column3 to Column6 are the same as those in Table 5. The last two columns are the differences between Column3, Column4 and Column5, Column6. We can observe that the differences except for the questions #450 and #453 are less than 8.0%. The small \( \Delta \text{ACC} \) and \( \Delta F_1 \) values validate that the classification models perform closely in \( S_{S,Q_i}^{10000} \) and \( S_{R,Q_i} \). That is, the distributions of the two datasets are similar. Therefore, we can safely use the generated data for the classification tasks where data privacy is concerned. It is worth noting that the accuracy values are generally higher than \( F_1 \) scores, but \( \Delta F_1 \) values are generally smaller than \( \Delta \text{ACC} \). This means that the Accuracy metrics are biased due to the class imbalance. Therefore, \( F_1 \) is more preferable to evaluate data with class imbalance.

4.4. Remarks

In addition, we conducted experiments using real data as a training set, and evaluated the classification performance using the resulting model (see Tables C.11 and C.12 in Appendix C). The results also confirmed that the proposed model is capable of...
generating data with the desired diversity, as shown in Figs. 7 and 8. Furthermore, we evaluated the model with a different training sample size to assess the impact (see Table C.13 in Appendix C). Comparing with Table 6 (results from 10,000 training samples), we can conclude that a larger training sample size can improve classification performance. This may be because the model can learn the distribution and diversity of larger training samples.

Comparing the accuracy in Table 5 and C.11, and the F1 scores in 6, C.12, and C.13, we can see that the Accuracy and F1 scores of the generated data decrease, but with a slightly different rate of decrease. This can be explained by the following reasons: First, WDCGAN limits the update of the generator parameters to a small range to achieve stable model convergence, which makes the generated data less diverse than the real data. The less diversity means that the classifiers trained with the generated data have less generalization. As a result, when the model is tested with real data, the accuracy and F1 scores can be much lower. However, if the generated data are used as a test set, the accuracy and F1 scores of the classifiers trained by the real data only slightly decrease. Second, the number of categories in the training data varies, resulting in significant differences in the data generated by the model. The more categories labeled in the training set, the more information the model can obtain from the external conditions. The external conditions allow the model to better learn the unique characteristics of the different data categories and help the model generate more categories. Last, since each classification model has different feature extraction abilities and sensitivities, this leads to significant differences in accuracy and the F1 score for different models on the same data.

In summary, through the above studies, we have seen that the proposed model, WDCGAN, can successfully generate realistic electricity consumption data, and the generated data can also be successfully applied to the classification tasks to identify sociodemographic information. The results validate the effectiveness of the proposed model.

5. Conclusions and future work

Data privacy and availability are becoming a growing concern in energy management. In this paper, we proposed an improved GAN model, WDCGAN, to generate realistic electric load profiles. The proposed model is derived from existing GAN variants, including DCGAN, CGAN, and WGAN. The WDCGAN improves the structure of the native GAN neural network to enhance its feature extraction capability, accelerate training convergence and stabilize the training process. The performance of the generative model is indirectly represented by the data it generates. To evaluate the proposed WDCGAN, we compared the generated data with the real data using the metrics including autocorrelation and RSME. In addition, we took the sociodemographic information identification task as an example to evaluate the performance of the generated data in real-world machine learning applications. The identification task was implemented based on the four classic classification models including CNN, LSTM, SVM, and lightGBM. We conducted the experiments comprehensively and the results showed that the proposed data generator can produce realistic data similar to the real data in terms of data distribution, patterns, and classification performance. The results confirm that real data can be replaced by generated data in various applications to address data privacy or availability problems.

For future work, we would like to further improve the WDCGAN model, e.g., by enhancing its ability to simulate the diversity of the generated data. Furthermore, it would be interesting to apply a distributed approach to speed up the training process and generate scalable datasets in a distributed manner.

Credit author statement

Jianbin Li: Conceptualization, Writing - review & editing, Project administration, Funding acquisition. Zhiqiang Chen: Conceptualization, Methodology, Investigation and resources, Writing - original draft, Writing - review & editing. Long Cheng: Conceptualization, Methodology, Writing - review & editing. Xiufeng Liu: Conceptualization, Methodology, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was supported by National Key Research and Development Plan of China (No. 2021YFE0102400) and Fundamental Research Funds for the Central Universities (2021MS017).

Appendix A. Nomenclature

See Tables A.7, A.8, A.9 and A.10.

Table A.7
Functions

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>D(G(Z))</td>
<td>The judgment result by discriminator with the input as synthetic sample G(Z)</td>
</tr>
<tr>
<td>D(X)</td>
<td>The judgment result by discriminator with the input as real sample X</td>
</tr>
<tr>
<td>D</td>
<td>The discriminator in generative adversarial network</td>
</tr>
<tr>
<td>G(Z)</td>
<td>The synthetic sample produced by the generator under input Z</td>
</tr>
<tr>
<td>G</td>
<td>The generator in generative adversarial network</td>
</tr>
</tbody>
</table>

Table A.8
Indices

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>G_0(10000)</td>
<td>The generator trained with 10,000 real samples for Question i</td>
</tr>
<tr>
<td>G_o(40000)</td>
<td>The generator trained with 40,000 real samples for Question i</td>
</tr>
<tr>
<td>Q_i</td>
<td>The sociodemographic Question number in the CER survey</td>
</tr>
<tr>
<td>w_j</td>
<td>The parameters to be learned in the j-th classification model</td>
</tr>
<tr>
<td>X_i</td>
<td>The i-th sample in the set X</td>
</tr>
<tr>
<td>y_{ij}</td>
<td>The j-th characteristics of the i-th customer</td>
</tr>
<tr>
<td>Y_i</td>
<td>The j-th characteristics of all sociodemographic characteristics</td>
</tr>
</tbody>
</table>

Table A.9
Sets

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_0(i)</td>
<td>The dataset of 40,000 data sampled from S_0 for Question i</td>
</tr>
<tr>
<td>S_0</td>
<td>The cleaned data of 300,000 CER records</td>
</tr>
<tr>
<td>S_0gi(10000)</td>
<td>The synthetic dataset by Generator G_0(i)</td>
</tr>
<tr>
<td>S_0g_0(40000)</td>
<td>The synthetic dataset by Generator G_0(40000)</td>
</tr>
<tr>
<td>X</td>
<td>Real electricity consumption dataset</td>
</tr>
<tr>
<td>Y</td>
<td>The labels set of X</td>
</tr>
<tr>
<td>Z</td>
<td>The random dataset</td>
</tr>
</tbody>
</table>
Appendix B. Figures

See Figures B.10, B.11 and B.12.

![Fig. B.10. Preprocessing data format from $336 \times 1$ to $7 \times 48$](image-url)
Appendix C. Tables

Tables C.11 and C.12 show the accuracy and the $F_1$ scores when real data $S_{R, Qi}$ is used as the training set. Column2 to Column5 are results with the real data $S_{R, Qi}$ as test set; while Column6 to Column9 are the results with the generated data $S_{S, Qi}(10000)$ as the test set. The last four columns are the differences of the results.

Table C.11

<table>
<thead>
<tr>
<th>Question No.</th>
<th>Accuracy (Real as Train Real as Test)</th>
<th>Accuracy (Real as Train Generated as Test)</th>
<th>ΔACC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CNN</td>
<td>SVM</td>
<td>LSTM</td>
</tr>
<tr>
<td>300</td>
<td>70.4</td>
<td>67.8</td>
<td>68.3</td>
</tr>
<tr>
<td>310</td>
<td>76.6</td>
<td>69.3</td>
<td>72.9</td>
</tr>
<tr>
<td>401</td>
<td>58.1</td>
<td>52.5</td>
<td>53.8</td>
</tr>
<tr>
<td>410</td>
<td>78.4</td>
<td>81.2</td>
<td>81.4</td>
</tr>
<tr>
<td>450</td>
<td>62.2</td>
<td>58.6</td>
<td>63.3</td>
</tr>
<tr>
<td>453</td>
<td>66.3</td>
<td>57.8</td>
<td>62.1</td>
</tr>
<tr>
<td>460</td>
<td>52.2</td>
<td>48.3</td>
<td>49.1</td>
</tr>
<tr>
<td>4704</td>
<td>70.7</td>
<td>69.8</td>
<td>73.6</td>
</tr>
<tr>
<td>4905</td>
<td>57.2</td>
<td>52.9</td>
<td>54.0</td>
</tr>
<tr>
<td>6103</td>
<td>63.6</td>
<td>64.1</td>
<td>62.7</td>
</tr>
</tbody>
</table>
Table C.12 shows the results of using 40,000 real or generated samples to train the model, $G_{0}(40000)$, but test with the real data.

<table>
<thead>
<tr>
<th>Question No.</th>
<th>Class No.</th>
<th>F1-Score (Real as Train, Real as Test)</th>
<th>F1-Score (Generated as Train, Real as Test)</th>
<th>$\Delta F1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>3</td>
<td>CNN 55.3, SVM 54.1, LSTM 52.6</td>
<td>CNN 54.7, SVM 53.7, LSTM 54.2</td>
<td>0.3</td>
</tr>
<tr>
<td>401</td>
<td>2</td>
<td>CNN 55.3, SVM 48.5, LSTM 50.4</td>
<td>CNN 54.6, SVM 49.6, LSTM 51.2</td>
<td>0.6</td>
</tr>
<tr>
<td>410</td>
<td>2</td>
<td>CNN 72.4, SVM 72.4, LSTM 72.7</td>
<td>CNN 72.4, SVM 71.4, LSTM 68.9</td>
<td>0.0</td>
</tr>
<tr>
<td>450</td>
<td>3</td>
<td>CNN 52.5, SVM 45.3, LSTM 60.8</td>
<td>CNN 52.6, SVM 46.1, LSTM 50.7</td>
<td>0.0</td>
</tr>
<tr>
<td>453</td>
<td>3</td>
<td>CNN 50.3, SVM 50.1, LSTM 62.2</td>
<td>CNN 48.7, SVM 47.0, LSTM 50.3</td>
<td>0.6</td>
</tr>
<tr>
<td>460</td>
<td>2</td>
<td>CNN 43.7, SVM 41.2, LSTM 45.3</td>
<td>CNN 44.4, SVM 37.8, LSTM 38.8</td>
<td>0.7</td>
</tr>
<tr>
<td>47004</td>
<td>2</td>
<td>CNN 50.6, SVM 55.0, LSTM 60.8</td>
<td>CNN 69.6, SVM 55.3, LSTM 51.2</td>
<td>0.4</td>
</tr>
<tr>
<td>4905</td>
<td>2</td>
<td>CNN 51.2, SVM 42.7, LSTM 53.1</td>
<td>CNN 51.2, SVM 47.3, LSTM 52.2</td>
<td>0.0</td>
</tr>
<tr>
<td>6103</td>
<td>2</td>
<td>CNN 56.1, SVM 51.3, LSTM 52.9</td>
<td>CNN 55.4, SVM 51.1, LSTM 47.5</td>
<td>0.7</td>
</tr>
</tbody>
</table>

References


