IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 23, 2020, accepted November 5, 2020, date of publication November 10, 2020,
date of current version November 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3037063

Conditional Tabular GAN-Based Two-Stage Data Generation Scheme for Short-Term Load Forecasting

JAEUK MOON, (Graduate Student Member, IEEE), SEUNGWON JUNG, (Graduate Student Member, IEEE), SUNGWOO PARK, (Graduate Student Member, IEEE), AND EENJUN HWANG, (Member, IEEE)
School of Electrical Engineering, Korea University, Seoul 02481, South Korea

Corresponding author: Eenjun Hwang (ehwang04@korea.ac.kr)

This work was supported in part by the Korea Electric Power Corporation under Grant R18XA05, and in part by the Energy Cloud Research and Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT under Grant 2019M3F2A1073184.

ABSTRACT Load forecasting is one of the critical tasks for enhancing the energy efficiency of smart grids. Even though recent deep learning-based load forecasting models have shown excellent forecasting performance, one of the common problems they faced was that their forecasting accuracy was highly dependent on the data quality and quantity available for the model training. Collecting a sufficient amount of high-quality data is expensive and time-consuming. Recently, a generative adversarial network (GAN) has shown its potential as a solution to the data shortage problem by generating virtual data based on a small amount of real data, and several studies have used GAN to generate electric load data for training forecasting models. However, due to the noise data problem of GANs, their predictive performance also deteriorated. To solve this problem, in this study, we propose a two-stage data generation scheme that more effectively generates input and output variables for short-term load forecasting. In the first stage, we generate virtual calendar and temperature data used as input variables using a conditional tabular GAN (CTGAN). In the second stage, we generate electric load data corresponding to the input variables using a deep learning-based regression model. Lastly, we construct our forecasting model by training another regression model using a mixture of generated data and real data. To verify the effectiveness of our scheme, we conducted extensive experiments using various datasets and data generation models. We report some of the results.

INDEX TERMS Short-term load forecasting, smart grid, conditional generative adversarial networks.

I. INTRODUCTION

A smart grid is a novel electrical power grid that combines information and communication technologies with the existing power grid. It aims to enhance energy efficiency by exchanging information in real-time between power suppliers and consumers. The smart grid enables the suppliers to precisely forecast the electrical power demand by providing them with information, such as current energy consumption and user profile. As a result, suppliers can maximize efficiency by generating the right amount of electricity. The smart grid has attracted much attention due to its diverse advantages, and many studies have been conducted to deal with various issues of the smart grid.

One critical issue of the smart grid is the precise prediction of electric loads that will be demanded during a specific period [1], [2]. Suppliers determine the amount of power that should be generated based on the prediction. If suppliers produce more electricity than necessary due to incorrect prediction, they will suffer from economic losses. In the opposite case, they will experience serious problems such as frequency drops and blackouts.

Electric load forecasting models are generally classified by the forecasting period [3]. In particular, short-term load forecasting (STLF) has been the key issue in smart grids because its short forecasting period is required for smart grid operation.

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Wang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

One popular approach for STLF is to use machine learning models, such as support vector regression (SVR) [4] and gradient boosted regression trees [5]. Those models capture diverse relevant data such as time, weather, and user profile through input variables and forecast future electric loads via output variables. More recently, with the development of deep learning technology, various prediction models based on deep learning have been proposed to achieve better forecasting accuracy [6], [7]. Even though such deep learning models have achieved significant performance improvement, their performance heavily depends on the quality and quantity of the historical data used for the model training. If the amount of historical data is insufficient or the data contains significant noise, their predictive performance will drop significantly due to improper learning.

On the other hand, collecting a sufficient amount of high-quality data is expensive and time-consuming. One popular method to overcome this problem is to complement insufficient data through artificial data generation. For instance, Li *et al.* [8] combined a mega-trend-diffusion technique with data trend estimation to generate artificial data. Xu *et al.* [9] adapted a variational auto-encoder to tabular data by modifying the loss function and training two neural networks using evidence lower-bound loss [10]. The emergence of a generative adversarial network (GAN) [11] has accelerated that trend. Since its introduction in 2014, many variations have been proposed to improve its ability to generate data [12], [13], and have made outstanding achievements in diverse fields [14], [15]. For instance, GAN was used to generate data for training electric load forecasting models. There are two typical approaches for learning-based electric load forecasting: (1) time-series forecasting, in which only the variables to be predicted are considered, and (2) label variable forecasting through external variables corresponding to features. In the first approach, GAN was effective in generating electric load data [16], [17]. However, in the second approach, GAN sometimes failed to learn the correlation between variables and showed poor forecasting accuracy [18], [19]. For example, although the generated input variables are similar to the real input variables, having a high value of output variables, they sometimes have a low value of output variables. GAN generally regards each variable dimension equally, regardless of whether they are input or output. This leads to ignoring the importance of output variables and generating incorrect data points for output variables.

To alleviate this problem, in this paper, we propose a two-stage data generation scheme to construct a large dataset for training a short-term load forecasting model. The dataset can be divided into two categories: external feature data and electric load data. In the first stage, we train a conditional tabular GAN (CTGAN) [9] using real external feature data such as calendar and temperatures and generate artificial feature data using the trained CTGAN. Because STLF variables can be described as tabular data, CTGAN, which can model tabular data distributions well, is suitable for generating data for STLF models. In the second stage, we generate electric load data using a deep learning-based regression model, and then label the input variables generated in the first stage. Finally, we construct our STLF model by training another deep learning-based regression model using a mixture of real and generated data. To evaluate the performance of our scheme, we conducted extensive experiments using various datasets and data generation schemes and measure the prediction accuracy.

The contributions of this paper are as follows:

- We utilize CTGAN to generate external feature data for STLF models. To the best of our knowledge, this is the first use of CTGAN for electric load forecasting.
- We propose a two-stage data generation scheme that generates external feature data and electric load data separately to represent their correlation more effectively.
- We compare the performance of various data generation schemes in the domain of electric load forecasting.

The remainder of this paper is organized as follows: Section 2 presents the literature review; Section 3 briefly summarizes the datasets; Section 4 presents the details of the proposed scheme; Section 5 demonstrates various experiments; and finally, Section 6 presents the conclusions.

II. BACKGROUNDS

A. RELATED WORKS

This section presents a brief literature review on machine learning-based load forecasting, GANs, and GAN-based forecasting. Among the various approaches for load forecasting, load forecasting using machine learning algorithms has attracted much attention recently. For instance, Fard and Akbari-Zadeh [20] proposed a hybrid method based on wavelet transform, artificial neural network (ANN), and autoregressive integrated moving average for STLF. They used autocorrelation and partial autocorrelation functions to observe the stationary and non-stationary behaviors of an electric load time series and determined the model configuration based on the behavior. Grolinger *et al.* [21] proposed STLF models based on SVR and ANN and discussed the strengths and weaknesses of each model by comparing them in the diverse experimental environments, including datasets and granularities (daily, every 15 minutes, etc.). They also presented a model selection algorithm to determine the hyperparameters of SVR and ANN.

More recently, many deep learning-based models have been proposed for STLF. For instance, Kuo and Huang [22] proposed a CNN-based STLF model, where the input variables were the previous electric loads and the output variables were future electric loads. They reported that their model was more accurate than five comparable artificial intelligence methods; SVR, random forest, decision tree, multi-layer perceptron (MLP), and long short-term memory (LSTM). Shi *et al.* [23] proposed a pooling-based deep recurrent neural network (RNN) that batches a set of customer load profiles into a pool of input. Ekonomou *et al.* [24] implemented an ANN-based STLF model reinforced by the appropriate

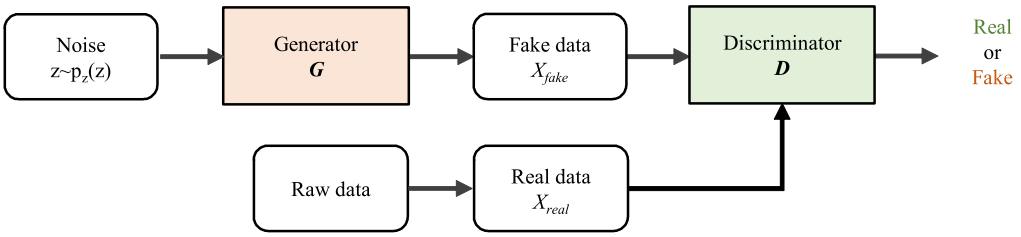

FIGURE 1. Architecture of vanilla GAN.

wavelet analysis. Tian *et al.* [25] integrated the hidden features of CNN and LSTM models to improve forecasting accuracy. With the CNN, they extracted the local trend of electric loads and captured similarities in usage patterns. They then trained the LSTM model to learn the relationship between the trend and the pattern with respect to time.

When GAN was introduced, it drew much attention due to its novel theory and performance. However, training GAN is not an easy task because GAN is sensitive to its hyperparameters and often suffers from serious problems, such as failing to generate the given data or learning only a small portion of the data [26], [27]. Hence, several attempts have been made to stabilize GAN training and simultaneously improve GAN performance. For instance, Arjovsky *et al.* [12] proposed Wasserstein GAN (WGAN) by introducing a Wasserstein distance to the loss functions of GANs. Gulrajani *et al.* [13] applied the gradient penalty to WGAN (WGANGP) to further improve the WGAN performance.

GANs have been widely used for diverse purposes and for solving forecasting tasks. Accordingly, several GAN-based studies have also been reported. Generative models were used for forecasting in the following studies. Tian *et al.* [16] proposed a parallel prediction scheme using a small number of original data to generate artificial data using a GAN. They formed a mixed dataset that included both original and artificial data and utilized it to train machine learning models for forecasting. Zhang *et al.* [28] proposed a GAN model for stock market forecasting and constructed their model based on an LSTM and an MLP. They used daily stock data to train the model and succeeded in achieving improved forecasting accuracy. Rezagholiradeh and Haidar [29] utilized GAN to resolve a regression problem. They modified the GAN structure such that their GAN model can simultaneously generate training data and perform forecasting. Consequently, their model succeeded in error reduction.

B. GAN

GAN [30] (Fig. 1) is a generative model based on the idea of game theory, which learns data distributions through an adversarial training process. The GAN is composed of two networks: generator G and discriminator D. The generator aims to generate realistic data with a distribution similar to that of real data and deceive the discriminator into discriminating them as real data. On the contrary, the discriminator aims to precisely distinguish whether input data come from the generator or real data. Such conflicting goals lead to simultaneous network training through competition. Equation (1) formulates such a process.

$$\mathrm{V}\,(\mathrm{D},\,\mathrm{G}) = E_{x\sim p_r}\left[\log D\,(x)\right] + E_{z\sim p_z(z)}\left[\log\left(1-D\,(G(z))\right)\right] \tag{1}$$

Here, x represents the data drawn from the real data distribution p_r, p_z is the noise distribution, z is the noise from p_z, $G(z)$ refers to the data generated by the generator with z as an input, and D indicates the discriminator function that outputs 1 if the given data are real; otherwise, it outputs 0. The discriminator attempts to maximize the value function $V(D, G)$ because $D(x)$ becomes 1, and $D(G(z))$ becomes 0 if $V(D, G)$ reaches the maximum. Conversely, the generator attempts to minimize $V(D, G)$ to ensure that $D(G(z))$ becomes 1. Even though the generator and discriminator hardly achieve their own goals during GAN training because of their competitive dynamic, the generator can generate realistic data at the end of the training.

III. MODEL ARCHITECTURE

In this section, we describe the architecture of our model. Our model is composed of two main parts: (i) a data generation part that generates external data and electric load data for training our forecasting model, and (ii) a regression-based forecasting model that is trained using the real data and generated data. Data generation has two modules: CTGAN for generating external feature data such as temperature and calendar data and a deep learning-based regression model for generating electric load data corresponding to the external feature data.

A. DATA GENERATION MODEL: CTGAN

Even though vanilla GAN has proven its superior ability to generate data, it cannot fully generate every type of data. For instance, vanilla GAN can hardly model all distributions in tabular data when the independent variables in the table have different distributions [31]. In addition, if the given data include categorical information represented by a one-hot vector, vanilla GAN cannot ensure the properties of the one-hot vector.

CTGAN [9] was proposed to solve such problem through two novel techniques. The first technique is the mode-specific

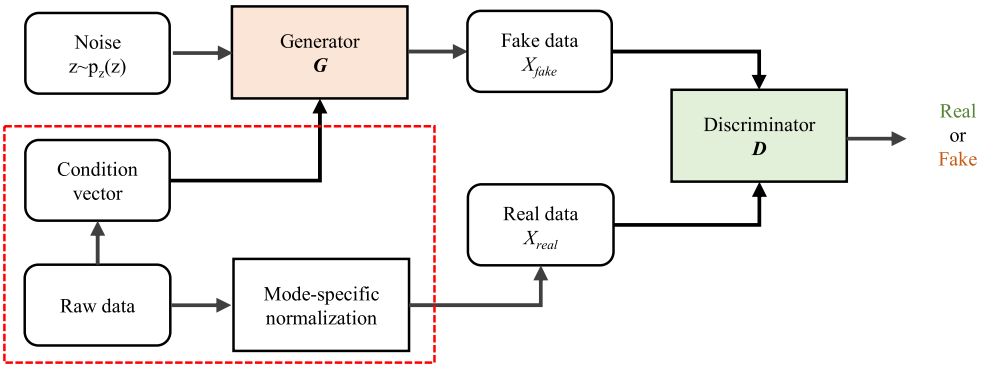

FIGURE 2. Architecture of conditional tabular GAN (CTGAN).

normalization N (0, 1) for a continuous variable consisting of float values. CTGAN uses a variational Gaussian mixture model [32] as a unit distribution model to estimate the number of distributions for each variable and normalize the variable values according to the estimated distributions. Then, CTGAN utilizes these encoded values in place of the original values during training. When generating artificial data after the training, CTGAN transforms the generated data into the original scale. The second technique is a conditional training approach for handling imbalanced category-level frequencies in categorical variables. The frequency imbalance makes the GAN generator produce only a few categories that frequently appear in the given data. Hence, this problem should be resolved to generate various data. The conditional training approach addresses this problem as follows: each column of the tabular data and the categorical variables are encoded into condition vectors. These vectors are sampled according to the log-frequency of the categories to ensure that rare categorical levels are evenly sampled. These vectors are then used as generator inputs. In addition to these techniques, CTGAN also leverages recent advances in GAN training, such as loss function of the WGANGP [13] and discriminator architecture of the PacGAN [33], which improve both the training stability and quality of the generated data. Fig. 2 shows its structural difference from the vanilla GAN and Equation (2) presents the loss function of CTGAN. In the equation, the first two terms indicate the original loss of the WGAN [12], and the last term indicates the gradient-penalty loss to control the gradient of the discriminator for random samples $\hat{x} \sim P_{\hat{x}}$ [13]. P_r and P_g represent the distribution of real and generated data. Here, λ is a gradient penalty coefficient and $\hat{x}$ represents samples that are linearly interpolated by real data x.

$$\mathrm{L} = E_{G(z)\sim P_g}\left[D(G(z))\right] - E_{x\sim P_r}\left[D(x)\right] + \lambda E_{\hat{x}\sim P_{\hat{x}}}\left[\left(\left\|\nabla_{\hat{x}} D(\hat{x})\right\| - 1\right)^2\right] \quad (2)$$

B. VARIABLE CONFIGURATION

We constructed data containing 10 input variables and one output variable. The output variable was an electric load during a specific period. The input variables included diverse data relevant to the output variable such as time and temperature, which have been popularly used in electric load forecasting [34], [35]. Table 1 lists the input and output variables that we considered in this paper.

TABLE 1. Input and output variables for model training.

Input variables	Description	Variable type
$Month_x$	Sine value at the month	Continuous on [−1,1]
$Month_y$	Cosine value at the month	Continuous on [−1,1]
Day_x	Sine value at the day	Continuous on [−1,1]
Day_y	Cosine value at the day	Continuous on [−1,1]
$Hour_x$	Sine value at the hour	Continuous on [−1,1]
$Hour_y$	Cosine value at the hour	Continuous on [−1,1]
$Weekday_x$	Sine value at the weekday	Continuous on [−1,1]
$Weekday_y$	Cosine value at the weekday	Continuous on [−1,1]
Holiday	Weekdays/holidays status	Binary [1:holiday, 0:others]
Temp.	Temperature value	Continuous
Output variable	**Description**	**Variable type**
Electric load	Hourly electric load	Continuous

The time information consists of month, day, hour, and weekday, and whether that date is a holiday or not. Month, day, hour, and weekday are numeric values initially; for instance, month and hour range from 1 to 12 and from 1 to 24, respectively. As these numeric values cannot reflect periodic properties, we transformed them into continuous data (i.e., $time_x$, and $time_y$) in a two-dimensional space using Equations (3) and (4) to represent their periodicity [6].

$$time_x = \sin\left((360/cycle) \times time\right) \quad (3)$$

$$time_y = \cos\left((360/cycle) \times time\right) \quad (4)$$

Here, *cycle* represents the period of time. For instance, if we transform month=1 into the $month_x$ and $month_y$, time and cycle become 1 and 12, and then, $month_x$ and $month_y$ become $sin(30)$ and $cos(30)$, respectively. Further explanations of this transformation can be found in this paper [36].

We also used a binary variable of either 0 or 1 to indicate whether that date is a holiday or not. Lastly, temperature is closely related to the operation of appliances, such as air conditioners and heaters, that require a lot of electricity [37]; therefore, this variable has been widely used in STLF models [38]–[40]. We also exploited this variable herein. However, the input variable configuration can be varied according to the applicability of data. For example, the past electric load data can be utilized as an input variable. Our scheme can achieve improved forecasting accuracy even in this condition.

C. FORECASTING MODEL

Now, we describe the overall steps for constructing our STLF model. Fig. 3 shows the overall architecture of our scheme, and the numbers (1)–(6) in the figure represent the flow of the main steps in our scheme.

First, (1) we train a CTGAN using input variables X_{real} and (2) generate artificial input variables X_{fake}. As the CTGAN learns the characteristics of X_{real} via this training, it can generate variables similar to X_{real}. Nine variables, including $Month_x$, $Month_y$, and Temperature, are used as continuous variables, which require mode-specific normalization. We follow the conditional training approach for the remaining variable (i.e., Holiday).

Second, we generate the artificial output variable Y_{fake} corresponding to X_{fake}. To do this, (3) we train a regression model *Reg_out* using X_{real} and their correspondent output variable Y_{real} so that *Reg_out* can predict the output variable when the input variables are given. When X_{fake} is given as an input variable, *Reg_out* forecasts the estimated output variable. (4) We make *Reg_out* predict for each X_{fake} and denote the obtained output variable by Y_{fake}. Here, any regression model (e.g., random forest, linear regression, and gradient boosting machine) can be used as *Reg_out*. However, we only consider deep learning-based regression models because we focus on generating sufficient data for deep learning models.

Finally, (5) we develop a training dataset by combining X_{real}, Y_{real}, X_{fake}, and Y_{fake} and (6) train another regression model *Reg_stlf* using the dataset to construct our STLF model.

The reason we generate X_{fake} and Y_{fake} separately using CTGAN and *Reg_out* is to reduce the possibility for the CTGAN to generate wrong output variable. Each data dimension generally has the same weight in the CTGAN. Thus, the relative weights of the input and output variables are determined by the number of their dimensions. In most cases, the dimension of the input variables is larger than that of the output variables; hence, the CTGAN gives more attention to the input variables than the output variables. Hence, if the CTGAN generates input and output variables simultaneously, generated output variables are not well-matched with the input variables.

Several studies on semi-supervised learning have investigated a similar approach to this learning scheme [41], [42]. For instance, in the pseudo-label [43] scheme, a classification model is trained with the labeled data through supervised learning; the unlabeled data are then labeled by the trained model and the generated labels are regarded as true labels in the remaining training steps.

After all, the pseudo-label generates labels for unlabeled real data based on labeled real data. On the contrary, our scheme generates both data and labels artificially based on a small amount of real labeled data. Hence, the performance of our scheme heavily depends on the quality of the generated data.

Algorithm 1 Data Generation and Model Construction

Input: A set of real input variables: $X_{real} = \{x_1, x_2, x_3, \cdots, x_n\}$,
a set of real output variables: $Y_{real} = \{y_1, y_2, y_3, \cdots, y_n\}$
Output: A trained regression model *reg_stlf*
Set of artificial input variables: $X_{fake} = \{x'_1, x'_2, x'_3, \cdots, x'_{n'}\}$
Set of artificial output variables: $Y_{fake} = \{y'_1, y'_2, y'_3, \cdots, y'_{n'}\}$
Train CTGAN using X_{real}
Generate X_{fake} using the trained CTGAN where $n' = 2n$
Train a regression model *reg_out* using X_{real} and Y_{real}
Generate Y_{fake} using the trained *reg_out* for input X_{fake} where $n' = 2n$
Mix X_{real} and X_{fake} and then Y_{real} and Y_{fake}
Train a regression model *reg_stlf* using the mixed data

IV. EXPERIMENTAL SETUP

To evaluate the effectiveness of our scheme, we performed extensive experiments. Before we present the details of our experiments, we first introduce the datasets, comparison models, and evaluation metrics.

A. DATASET DESCRIPTION

For the experiment, we considered the following three electric load datasets.

- **GEFCOM 2012**: The first dataset includes hourly electric load data of a US utility and the temperature data from January 1, 2005 to December 31, 2008 for 20 zones, which were used in the load forecasting track of the Global Energy Forecasting Competition 2012 (GEFCom 2012) hosted on Kaggle [44]. This dataset is open to the public 45], and we used the data from 2007 to 2008. Out of the 20 zones, we only used data from zone_1 to zone_11 because the remaining nine zones had no temperature data.
- **Mendeley Malaysia**: The second dataset includes the hourly load data of the power supply company of Johor City in Malaysia, and the temperature data—ranging from January 1, 2009 to December 31, 2010—were provided by Mendeley [46]. This dataset is also open to the public. However, it does not contain any information

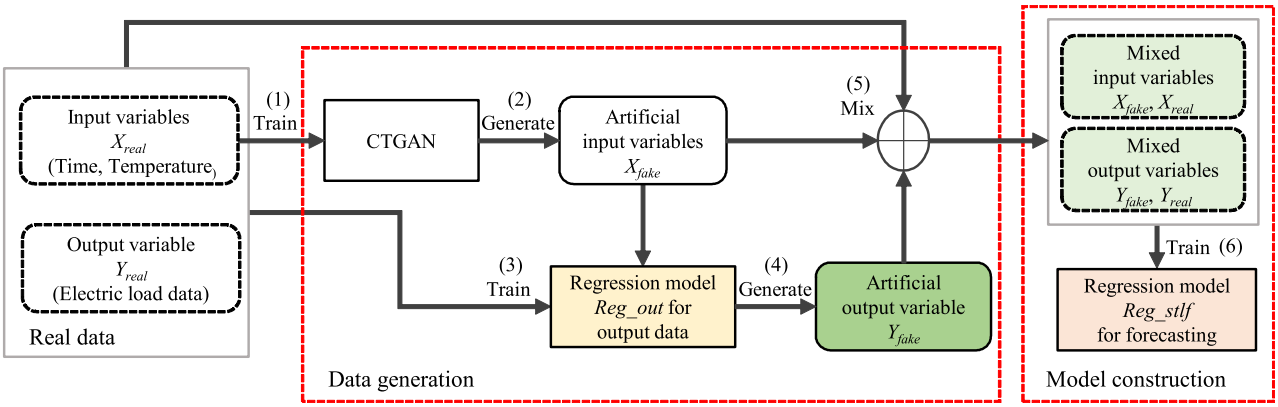

FIGURE 3. Overall structure of the proposed scheme.

on Malaysian holidays; thus, we collected the holiday information for the period from the Internet.

- **Private University**: The last dataset contains the hourly electric load data of private university buildings located in Seoul, South Korea, from January 1, 2015 to December 31, 2018. Among them, we used the data of the recent two years. We considered three types of school buildings: academic buildings, laboratory buildings, and dormitories. An academic building contains classrooms, department offices, and other administrative offices. Unlike other datasets, this dataset provides the load data only. Thus, we collected the temperature data from the Korea Meteorological Administration (KMA) and the holiday information from the Internet.

Based on these datasets, we constructed a total of fifteen datasets for the experiment: eleven datasets from the first dataset, one from the second dataset, and three from the third dataset. All dataset variables were normalized into a range of 0 to 1. Table 2 shows those datasets and some of their characteristics.

TABLE 2. Characteristics of fifteen datasets.

No.	Dataset	Public data
1	zone_1	O
2	zone_2	O
3	zone_3	O
4	zone_4	O
5	zone_5	O
6	zone_6	O
7	zone_7	O
8	zone_8	O
9	zone_9	O
10	zone_10	O
11	zone_11	O
12	Malaysia	O
13	Academic	X
14	Dormitory	X
15	Laboratory	X

B. MODEL SETUP

For performance comparison, we considered five other generation models: three GAN-based models, one statistical generation model, and one deep neural network generation model. The three GAN-based models are vanilla GAN [30], CGAN [47], and WGANGP [13], and the statistical and deep neural network generation model are Mega-trend-diffusion (MTD) [5] and Tabular Variational Auto Encoder (TVAE) [9], respectively. We implemented all these models using Python 3.7.3 with TensorFlow 1.13.1 and Pytorch 1.5.0.

We used similar hyperparameters for the GAN-based models. The generators and discriminators were set to have three layers with seven perceptrons each, and sigmoid and rectified linear unit (ReLU) were used as their activation function, respectively. The number of epochs, learning rate, noise dimension and batch size were 1000, 0.001, 20 and 50, respectively. In the case of CGAN, as CGAN requires both inputs and an associated condition unlike other GAN-based models, we used normalized electric loads as its condition when we train the model. Furthermore, when generating artificial input variables, we used a condition with an even distribution over 0 to 1 for uniformly distributed output variables. That is, the CGAN condition was set to 0, 0.1, 0.2, . . . , 1.0. For other hyperparameters not mentioned above, we followed the experimental setup described in each model's paper. For the input variable column designation of the CTGAN, we set the "*Holiday*" variable as the categorical column and the other variables as continuous columns. For the other hyperparameters of the CTGAN and TVAE, such as optimizer, activation, and the number of epochs, we followed the default settings of SDGym [48]. In the case of MTD, we implemented the original paper [5].

We used MLP as the regression model, the hyperparameters of which were unchanged throughout the experiment. For implementation, we used scikit-learn library [49]. We set the number of hidden layers to seven, empirically. Furthermore, we set the number of perceptrons in each hidden layer to

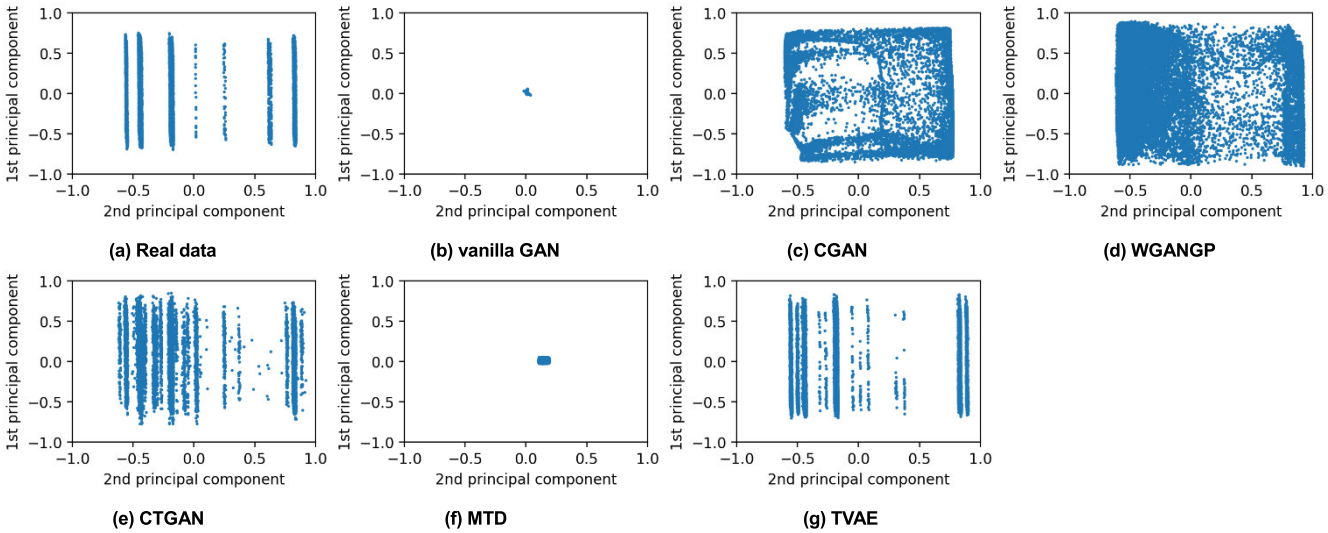

FIGURE 4. PCA of real data and input variables generated by various generation models.

seven, which was approximately 2/3 of the number of input variables according to [35]. The objective function was a mean squared error between forecasting results and true electric loads. The other settings followed the default settings of the library. We only used 10-dimensional input variables at one point for forecasting; thus, we did not experiment with other deep learning models that require a sequence of data input variables at a consecutive time (e.g., CNN and RNN).

C. EVALUATION METRICS

For the accuracy comparison, we calculated the mean absolute percentage error (MAPE), root mean square error (RMSE), and mean absolute error (MAE), which are the most popular metrics used for comparison. The MAPE, RMSE, and MAE are computed in Equations (5), (6), and (7), respectively, where N is the number of data samples, A_t is the actual electric load, and F_t is the forecasted value electric load.

$$\text{MAPE} = 100 \times \frac{1}{N} \sum_t \left| \frac{A_t - F_t}{A_t} \right| \tag{5}$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_t (A_t - F_t)^2} \tag{6}$$

$$\text{MAE} = \frac{1}{N} \sum_t |A_t - F_t| \tag{7}$$

V. EXPERIMENTS AND DISCUSSION

In this section, we present four experiments that we performed to evaluate the effectiveness of our scheme. First, we investigate the effectiveness of various data generation schemes. Second, we evaluate the performance of the forecasting models trained using those generated datasets. Third, we compare the effect of our separate data generation scheme and traditional data generation scheme, which generates input variables and output variables at the same time. Finally, we evaluate our scheme with a different input variable configuration. In all the experiments, we divided all datasets into training and test sets at a ratio of 5:5 (1 year each). Thus, the number of training and test data was 8,766 each. When we generate artificial data using diverse generation models, we generated 17,532 data samples, twice as much as the training data.

A. DATA GENERATION

In this experiment, we consider two types of data generation: one for input variables and the other for output variables. In the data generation for input variables, we considered six different data generation schemes and compared their effectiveness by using a principal component analysis (PCA) [50]. PCA is one of the visualization methods for representing multi-dimensional data by reducing the data dimension using principal components extracted from the given data. When extracting two principal components from the generated data, the generated data samples are represented in a two-dimensional space. In the data generation for the output variable, we used our regression model *Reg_out* to generate data corresponding to the input variables and visualized their distribution using a histogram. We conducted this analysis for all the datasets. For instance, Fig. 4 depicts the distribution of real data and input variables generated by vanilla GAN, CGAN, WGANGP, CTGAN, MTD, and TVAE, respectively, in terms of the first two principal components for the Mendeley Malaysia dataset. Fig. 5 depicts the histogram of real data and the output variables produced by *Reg_out* using the generated input variables. In the figures, the x-axis and y-axis represent the distribution of normalized output variables and the frequency of each output variable, respectively.

The data distribution in Fig. 4(a) shows several bars. The distributions of Figs. 4(b)∼(g) show three different patterns. In the case of vanilla GAN and MTD (Figs. 4(b) and (f), respectively), most of the generated input variables were located near the origin. That is, these models learned only the

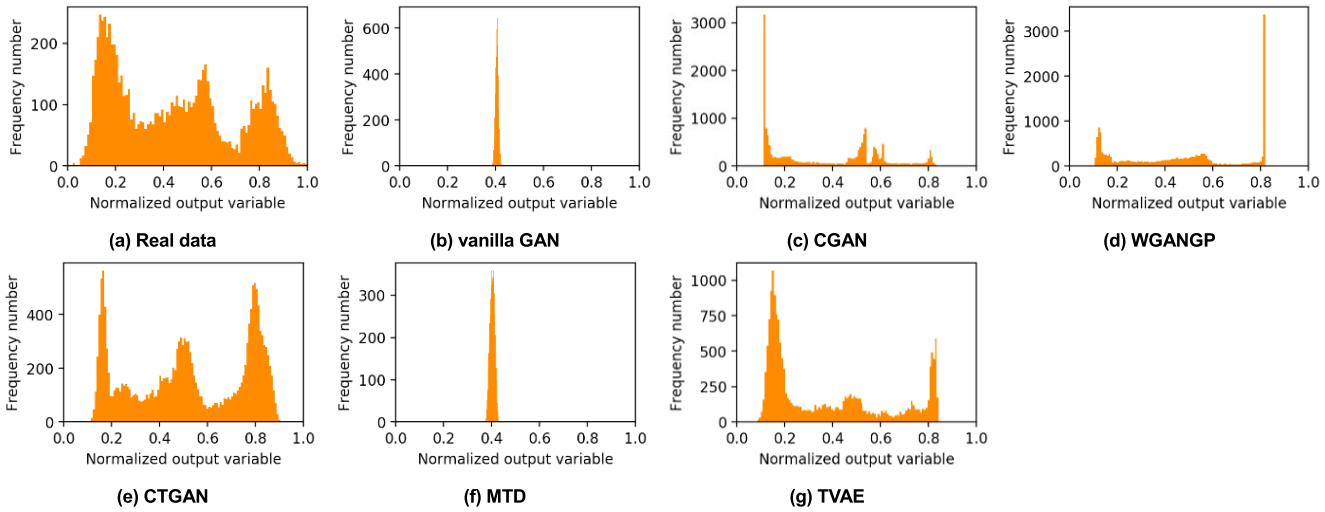

FIGURE 5. Distribution histogram of real data and the output variables generated by *Reg_out*.

TABLE 3. MAPE comparison of forecasting results.

Dataset	MLP	GAN	CGAN	WGANGP	CTGAN	MTD	TVAE
	Real-1yr	Real-1yr + Generated Data					
zone_1	14.71	14.09	13.35	13.22	**12.91**	13.09	14.24
zone_2	10.73	9.76	9.83	8.74	**8.31**	9.27	9.35
zone_3	8.47	9.55	8.44	8.36	**8.32**	9.74	8.65
zone_4	21.57	19.74	21.52	19.77	**18.22**	19.61	19.10
zone_5	14.77	15.46	14.58	15.30	**13.25**	17.33	13.94
zone_6	9.81	8.89	**8.47**	8.90	8.75	9.55	9.08
zone_7	8.72	8.35	8.89	8.59	**8.27**	8.54	8.45
zone_8	11.94	11.40	11.87	11.50	11.84	11.92	**11.32**
zone_9	176.28	171.67	167.79	**164.94**	170.61	169.25	174.72
zone_10	11.39	11.27	10.38	10.59	11.87	11.13	**10.02**
zone_11	14.07	13.85	13.72	13.62	**13.17**	13.98	13.80
Malaysia	12.75	10.78	11.42	11.74	**10.66**	10.70	12.11
Academic	15.89	16.58	15.22	16.35	**14.70**	18.43	16.10
Dormitory	11.07	10.48	10.60	10.64	**9.69**	10.29	10.29
Laboratory	10.92	10.62	9.83	11.73	**9.69**	10.25	10.69

mean or mode of the data distribution rather than the whole data distribution. As a result, their output variable distribution was concentrated around 0.4, as shown in Figs. 5(b) and (f). On the contrary, the input variables of CGAN and WGANGP (Figs. 4(c) and (d), respectively), had a square-shaped distribution including the distribution boundary of the real data. Hence, their output variables were spread between 0.1 and 0.8. Although WGANGP and CGAN generated some output variables over the entire range, they were still focused on specific areas. On the contrary, CTGAN and TVAE (Figs. 4(e) and (g), respectively) showed the most similar distribution of the input variables to the real data distribution. In addition, the distributions of their output variables were similar. In particular, CTGAN covered the largest portion among the generation models and the distribution was the most similar to the real data. The other datasets showed similar results.

B. FORECASTING PERFORMANCES

We generated artificial input variables using six different generation models for fifteen datasets in the first experiment to augment the dataset for training a forecasting model. Hence, to determine the effectiveness of data augmentation, we investigate the accuracy of the regression-based forecasting models constructed by using the six augmented datasets. As a baseline, we constructed an MLP, trained it using actual 1-year, and measured its forecasting accuracy. Tables 3, 4, and 5 show their MAPE, RMSE, and MAE, respectively. The values in the tables represent the average

TABLE 4. RMSE comparison of forecasting results.

Dataset	MLP	GAN	CGAN	WGANGP	CTGAN	MTD	TVAE
	Real-1yr	Real-1yr + Generated Data					
zone_1	4026.07	3974.62	3673.91	3866.79	**3664.96**	3949.87	3937.80
zone_2	25784.80	23551.15	23849.98	23289.12	**20053.93**	22629.85	22445.22
zone_3	22277.84	25046.28	22071.18	21869.86	**21814.20**	25228.29	22794.50
zone_4	88.01	79.48	87.46	77.24	**74.50**	79.57	75.05
zone_5	1424.99	1430.19	1363.16	1375.43	**1209.44**	1592.94	1300.18
zone_6	24682.13	22707.51	**21610.51**	22423.67	22764.25	24263.91	23119.44
zone_7	22854.04	22391.96	23657.02	22338.21	**22001.37**	22113.89	22460.02
zone_8	673.40	639.09	651.50	639.01	665.67	671.07	**632.10**
zone_9	21504.38	21251.80	21323.59	21376.54	**21277.15**	21751.77	21254.68
zone_10	4032.99	4013.97	**3606.94**	3695.33	4187.48	3918.41	3685.11
zone_11	22888.21	22633.30	22592.03	22030.51	**21750.47**	23399.16	22717.28
Malaysia	6970.90	6192.82	6537.09	6585.22	**5941.74**	6125.80	6774.74
Academic	486.40	519.78	508.27	509.25	**454.30**	432.94	465.12
Dormitory	203.34	192.25	197.67	196.62	181.80	**143.40**	191.14
Laboratory	379.08	373.87	346.52	406.95	**344.25**	361.18	359.18

TABLE 5. MAE comparison of forecasting results.

Dataset	MLP	GAN	CGAN	WGANGP	CTGAN	MTD	TVAE
	Real-1yr	Real-1yr + Generated Data					
zone_1	3031.98	2950.32	2746.32	2826.59	**2713.61**	2935.87	2943.12
zone_2	19833.00	18152.67	18386.71	17741.98	**15393.78**	17362.13	17253.05
zone_3	17004.50	19306.20	16893.39	16772.05	**16707.21**	19564.56	17324.26
zone_4	68.84	61.60	68.38	60.12	**57.08**	61.80	58.62
zone_5	1096.63	1112.99	1051.14	1079.81	**947.69**	1250.86	1008.26
zone_6	18994.53	17292.03	**16470.24**	17265.14	17226.36	18583.18	17752.64
zone_7	17498.91	16990.58	18003.14	17197.12	**16796.63**	16999.06	17121.02
zone_8	510.79	485.92	503.99	486.16	501.52	509.70	**482.46**
zone_9	15391.58	**15160.43**	15298.85	15369.72	15301.12	15606.48	15282.91
zone_10	3111.95	3082.83	2810.28	2878.69	3216.63	3030.44	**2783.96**
zone_11	17455.90	17195.23	17113.79	16550.20	**16482.46**	17622.00	17277.47
Malaysia	5112.49	4308.38	4552.60	4743.45	**4268.85**	4293.92	4991.68
Academic	366.04	391.70	367.01	380.32	**337.65**	432.94	353.19
Dormitory	154.23	146.25	148.90	148.43	**135.79**	143.40	144.37
Laboratory	283.02	277.91	255.34	304.50	**254.05**	266.39	274.04

value of the results of repeating the prediction 10 times. The generation schemes that showed the best performance in each dataset are marked in bold font.

In the case of MAPE, the zone_9 dataset showed extraordinary errors from 164% to 176%. This is because it had many entries with zero electric load. This could result in a severe penalty in calculating MAPE.

As we can see in the tables, our CTGAN exhibited the best performance among all the generation schemes. Compared to MLP using real 1-year data, our CTGAN improved the MAPE in the fourteen datasets and showed a marginal difference for the one dataset. WGANGP also enhanced the forecasting accuracy for twelve out of fifteen datasets. TVAE and CTGAN showed very similar forecasting performance in most datasets because they had a very similar distribution of generated data in the previous experiment.

Overall, the data generation schemes we considered contributed to enhancing the forecasting performance. In particular, CTGAN demonstrated the best performance for most datasets in terms of MAPE because it has a superior ability to learn the overall distribution of the real data and effectively generate data following the distribution.

Tables 4 and 5 show that using properly augmented data for training can enhance the forecasting performance in terms of RMSE and MAE compared to only using real 1-year data. In particular, our CTGAN achieved better RMSE and MAE than the MLP using the 1-year real data for all datasets except for zone_10. On the contrary, MTD and GAN, which learned

TABLE 6. Comparison of our scheme and traditional scheme.

Dataset	Our scheme			Traditional scheme		
	MAPE	RMSE	MAE	MAPE	RMSE	MAE
zone_1	**12.91**	**3664.96**	**2713.61**	19.78	4302.61	3543.07
zone_2	**8.31**	**20053.93**	**15393.78**	11.74	28312.65	21827.54
zone_3	**8.32**	**21814.20**	**16707.21**	12.41	31419.21	24585.39
zone_4	**18.22**	**74.50**	**57.08**	23.84	93.66	75.34
zone_5	**13.25**	**1209.44**	**947.69**	22.52	2033.69	1618.60
zone_6	**8.75**	**22764.25**	**17226.36**	13.25	29879.71	24143.75
zone_7	**8.27**	**22001.37**	**16796.63**	12.97	31748.84	25205.56
zone_8	**11.84**	**665.67**	**501.52**	18.17	849.02	690.21
zone_9	170.61	**21277.15**	**15301.12**	**160.80**	22354.16	17122.19
zone_10	**12.04**	**4292.89**	**3294.14**	16.22	5289.43	4239.09
zone_11	**13.17**	**21750.47**	**16482.46**	16.46	27592.26	20496.85
Malaysia	**10.66**	**5941.75**	**4268.85**	17.97	8626.29	7015.90
Academic	**14.70**	**454.30**	**337.65**	29.75	848.63	669.83
Dormitory	**9.69**	**181.80**	**135.79**	15.03	249.25	197.62
Laboratory	**9.69**	**344.25**	**254.05**	16.83	561.31	432.62

only the mean or mode of the real data distribution, showed the worst performance among the generation models.

C. COMPARISON WITH TRADITIONAL METHOD

In this experiment, we compare our two-stage data generation scheme with the traditional one-stage approach, where both input and output variables are generated simultaneously. For a fair comparison, we used CTGAN for simultaneous data generation. Table 6 presents its MAPE, RMSE, and MAE. Our proposed scheme demonstrated better accuracy than the traditional scheme in all datasets except zone_9, where the traditional scheme achieved slightly better MAPE than our proposed scheme. Still, its RMSE and MAE were worse than our proposed scheme.

To observe the effectiveness more closely, we compared the forecasting results of the proposed scheme and traditional scheme with actual electric load data. Fig. 6(a) shows the comparison result over about one month for the Mendeley Malaysia dataset, and Figs. 6(b) and (c) show the enlarged portion of some weekends and weekdays, respectively. Overall, our scheme predicts the overall trend and maximum and minimum values more accurately than the traditional scheme.

D. INPUT VARIABLE FLEXIBILITY EVALUATION

In this experiment, we demonstrate the flexibility of our scheme in the input variable configuration by using datasets to which several input variables were newly added. We constructed new datasets by merging 10 original input variables and 7 variables indicating the electrical usage data during the past 7 days; thus, we had 17 input variables. Then, we measured the forecasting performance of a baseline model (MLP trained with actual 1-year data) and our scheme. The other experimental settings were the same as the previous experiments. Table 7 presents the forecasting performance of MLP and our scheme. The forecasting results of our scheme that showed better performance in each metric are marked in bold font. Compared with the results organized in Tables 3, 4, and 5, the overall forecasting performances of both MLP and our scheme were improved because we used past electrical usage data as input variables. Even in this case, our scheme outperformed the baseline MLP, especially in terms of MAPE and MAE. This result means that our scheme can be a reasonable way for improving the forecasting performance regardless of what input variables are used.

VI. CONCLUSION

In this paper, we proposed a two-stage data generation scheme for short-term load forecasting to solve the data shortage problem in training a forecasting model. We employed CTGAN, which is a novel GAN-based model, to generate input variables and a deep learning-based regression model to generate output variables using the available small real data. We then trained an electric load forecasting model using actual and generated data. To determine the effectiveness of our data generation scheme, we implemented five other data generation schemes, constructed forecasting models using their data, and compared their performances in terms of MAPE, RMSE, and MAE.

The experimental results demonstrated that as CTGAN can model tabular data distributions well, it is appropriate for generating input variables for STLF. Also, our data generation scheme enhanced the forecasting accuracy more than other data generation schemes. In addition, from the comparison with the actual electric load data, our proposed scheme can achieve better performance than traditional one-stage data generation method.

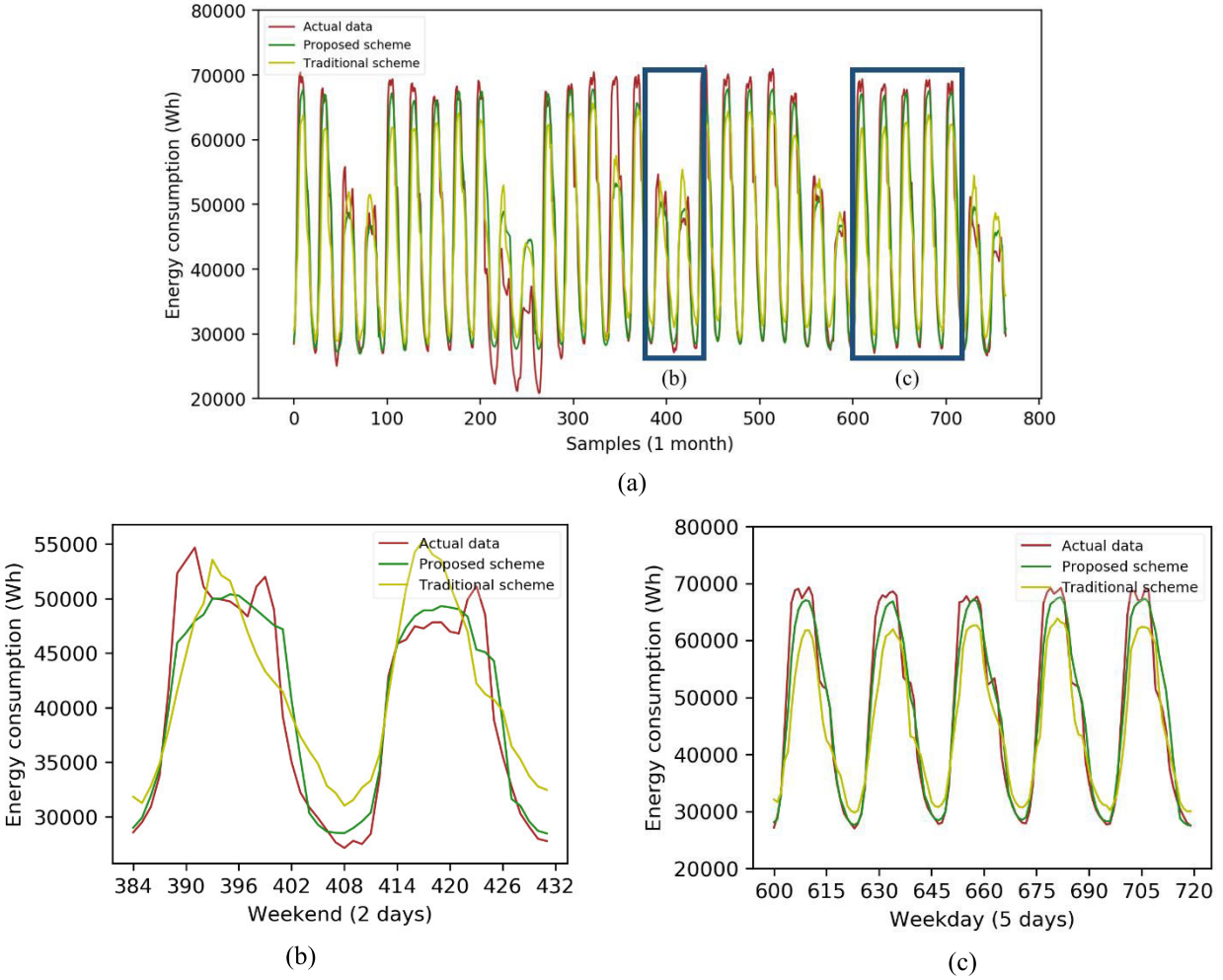

FIGURE 6. Comparison of 1-stage and 2-stage data generation schemes with actual data.

TABLE 7. Comparison of forecasting results with added input variable.

Dataset	MAPE		RMSE		MAE	
	MLP	Our scheme	MLP	Our scheme	MLP	Our scheme
zone_1	5.20	**2.96**	1379.99	**928.31**	1022.73	**628.56**
zone_2	5.15	**2.10**	11671.84	**10473.02**	9153.81	**4194.33**
zone_3	3.33	**1.80**	8746.455	**5222.26**	6571.91	**3637.85**
zone_4	28.91	**9.86**	125.30	**35.54**	100.15	**21.44**
zone_5	5.75	**3.78**	549.59	562.055	407.37	**308.87**
zone_6	3.30	**2.21**	8097.16	10451.16	6222.56	**4502.87**
zone_7	4.63	**1.79**	11632.49	**5273.60**	9128.59	**3638.58**
zone_8	5.87	**3.09**	319.31	**219.79**	238.15	**132.24**
zone_9	40.77	**24.32**	9962.76	**8469.49**	6235.72	**5325.16**
zone_10	5.55	**2.50**	2041.14	**1178.57**	1522.14	**718.87**
zone_11	5.42	**3.17**	8017.09	11594.39	6225.01	**4367.85**
Malaysia	5.47	**3.65**	3052.57	**2652.69**	2237.77	**1491.02**
Academic	11.53	**4.19**	299.72	**169.40**	225.00	**104.66**
Dormitory	5.63	**3.54**	99.00	**80.42**	75.25	**50.22**
Laboratory	5.65	**2.72**	204.77	**126.79**	146.15	**75.22**

However, the limitation of our scheme is that since our scheme is based on GAN, if the amount of data is so small, then the GAN cannot sufficiently train the distribution. GANs that are not sufficiently trained cannot be able to properly generate artificial data, which cannot be used to improve forecasting performance. In future works, we will make a

more applicable load forecasting scheme to such industry than this work by applying generation models which can train even in a small amount of data.

REFERENCES

[1] M. Son, J. Moon, S. Jung, and E. Hwang, "A short-term load forecasting scheme based on auto-encoder and random forest," in *Proc. Int. Conf. Appl. Phys., Syst. Sci. Comput. (APSAC)*. Dubrovnik, Croatia: Springer, Sep. 2018, pp. 138–144.

[2] J. Moon, J. Park, E. Hwang, and S. Jun, "Forecasting power consumption for higher educational institutions based on machine learning," *J. Supercomput.*, vol. 74, no. 8, pp. 3778–3800, Aug. 2018.

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

[4] A. Setiawan, I. Koprinska, and V. G. Agelidis, "Very short-term electricity load demand forecasting using support vector regression," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2009, pp. 2888–2894.

[5] J. Walther, D. Spanier, N. Panten, and E. Abele, "Very short-term load forecasting on factory level—A machine learning approach," *Procedia CIRP*, vol. 80, pp. 705–710, 2019, doi: 10.1016/j.procir.2019.01.060.

[6] J. Moon, S. Park, S. Rho, and E. Hwang, "A comparative analysis of artificial neural network architectures for building energy consumption forecasting," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 9, pp. 1–19, 2019.

[7] S. Park, J. Moon, S. Jung, S. Rho, S. W. Baik, and E. Hwang, "A two-stage industrial load forecasting scheme for day-ahead combined cooling, heating and power scheduling," *Energies*, vol. 13, no. 2, p. 443, Jan. 2020.

[8] D.-C. Li, C.-S. Wu, T.-I. Tsai, and Y.-S. Lina, "Using mega-trend-diffusion and artificial samples in small data set learning for early flexible manufacturing system scheduling knowledge," *Comput. Oper. Res.*, vol. 34, no. 4, pp. 966–982, Apr. 2007.

[9] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7335–7345.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114

[11] M. Son, S. Jung, J. Moon, and E. Hwang, "BCGAN-based over-sampling scheme for imbalanced data," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2020, pp. 155–160.

[12] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, *arXiv:1701.07875*. [Online]. Available: http://arxiv.org/abs/1701.07875

[13] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[14] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.

[15] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool, "Generative adversarial networks for extreme learned image compression," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 221–231.

[16] C. Tian, C. Li, G. Zhang, and Y. Lv, "Data driven parallel prediction of building energy consumption using generative adversarial nets," *Energy Buildings*, vol. 186, pp. 230–243, Mar. 2019.

[17] C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "Generative adversarial network for synthetic time series data generation in smart grids," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Oct. 2018, pp. 1–6.

[18] A. Torfi and E. A. Fox, "CorGAN: Correlation-capturing convolutional generative adversarial networks for generating synthetic healthcare records," 2020, *arXiv:2001.09346*. [Online]. Available: http://arxiv.org/abs/2001.09346

[19] Y. Pu, S. Dai, Z. Gan, W. Wang, G. Wang, Y. Zhang, R. Henao, and L. Carin, "JointGAN: Multi-domain joint distribution learning with generative adversarial nets," 2018, *arXiv:1806.02978*. [Online]. Available: http://arxiv.org/abs/1806.02978

[20] A. K. Fard and M.-R. Akbari-Zadeh, "A hybrid method based on wavelet, ANN and ARIMA model for short-term load forecasting," *J. Exp. Theor. Artif. Intell.*, vol. 26, no. 2, pp. 167–182, Apr. 2014.

[21] K. Grolinger, A. L'Heureux, M. A. M. Capretz, and L. Seewald, "Energy forecasting for event venues: Big data and prediction accuracy," *Energy Buildings*, vol. 112, pp. 222–233, Jan. 2016.

[22] P.-H. Kuo and C.-J. Huang, "A high precision artificial neural networks model for short-term energy load forecasting," *Energies*, vol. 11, no. 1, p. 213, Jan. 2018.

[23] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—A novel pooling deep RNN," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5271–5280, Sep. 2018.

[24] L. Ekonomou, C. Christodoulou, and V. Mladenov, "A short-term load forecasting method using artificial neural networks and wavelet analysis," *Int. J. Power Syst.*, vol. 1, pp. 64–68, Jul. 2016.

[25] C. Tian, J. Ma, C. Zhang, and P. Zhan, "A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network," *Energies*, vol. 11, no. 12, p. 3493, Dec. 2018.

[26] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," 2016, *arXiv:1612.02136*. [Online]. Available: http://arxiv.org/abs/1612.02136

[27] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.

[28] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, "Stock market prediction based on generative adversarial network," *Procedia Comput. Sci.*, vol. 147, pp. 400–406, Jan. 2019, doi: 10.1016/j.procs.2019.01.256.

[29] M. Rezagholiradeh and M. A. Haidar, "Reg-GAN: Semi-supervised learning based on generative adversarial networks for regression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 2806–2810.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[31] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, "VEEGAN: Reducing mode collapse in GANs using implicit variational learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3308–3318.

[32] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[33] Z. Lin, A. Khetan, G. Fanti, and S. Oh, "Pacgan: The power of two samples in generative adversarial networks," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 324–335, May 2020.

[34] J. Kim, J. Moon, E. Hwang, and P. Kang, "Recurrent inception convolution neural network for multi short-term load forecasting," *Energy Buildings*, vol. 194, pp. 328–341, Jul. 2019.

[35] J. Moon, S. Jung, J. Rew, S. Rho, and E. Hwang, "Combination of short-term load forecasting models based on a stacking ensemble approach," *Energy Buildings*, vol. 216, Jun. 2020, Art. no. 109921.

[36] J. Moon, J. Kim, P. Kang, and E. Hwang, "Solving the cold-start problem in short-tern forecasting using tree-based methods," *Energies*, vol. 13, no. 4, pp. 886–922, 2020.

[37] S. S. Reddy, "Bat algorithm-based back propagation approach for short-term load forecasting considering weather factors," *Electr. Eng.*, vol. 100, no. 3, pp. 1297–1303, Sep. 2018.

[38] B. Yildiz, J. I. Bilbao, and A. B. Sproul, "A review and analysis of regression and machine learning models on commercial building electricity load forecasting," *Renew. Sustain. Energy Rev.*, vol. 73, pp. 1104–1122, Jun. 2017.

[39] M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renew. Sustain. Energy Rev.*, vol. 50, pp. 1352–1372, Oct. 2015.

[40] L. Hernandez, C. Baladron, J. M. Aguiar, B. Carro, A. J. Sanchez-Esguevillas, J. Lloret, and J. Massana, "A survey on electric power demand forecasting: Future trends in smart grids, microgrids and smart buildings," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1460–1495, 3rd Quart., 2014.

[41] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3581–3589.

[42] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. R. Salakhutdinov, "Good semi-supervised learning that requires a bad GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6510–6520.

[43] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. Workshop Challenges Represent. Learn. (ICML)*, 2013, vol. 3, no. 2, pp. 1–6.

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

[45] Y. Wang, Q. Chen, T. Hong, and C. Kang, "Review of smart meter data analytics: Applications, methodologies, and challenges," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 3125–3148, May 2019.
[46] H. J. Sadaei, P. C. de Lima e Silva, F. G. Guimarães, and M. H. Lee, "Short-term load forecasting by using a combined method of convolutional neural networks and fuzzy time series," *Energy*, vol. 175, pp. 365–377, May 2019.
[47] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*. [Online]. Available: http://arxiv.org/abs/1411.1784
[48] X. Lei. (2020). *SDGym*. University Massachusetts Institute of Technology, Massachusetts, USA. [Online]. Available: https://github.com/sdv-dev/SDGym
[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[50] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

JAEUK MOON (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Korea University, Seoul, South Korea, in 2019, where he is currently pursuing the master's degree with the School of Electronic Engineering. His research interests include machine learning, deep learning, and time series forecasting.

SEUNGWON JUNG (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Korea University, Seoul, South Korea, in 2016, where he is currently pursuing the Ph.D. degree with the School of Electronic Engineering. His research interests include data mining, machine learning, deep learning, and time series forecasting.

SUNGWOO PARK (Graduate Student Member, IEEE) received the B.S. degree in computer engineering from Hongik University, Seoul, South Korea, in 2018. He is currently pursuing the Ph.D. degree with the School of Electronic Engineering, Korea University, Seoul. His research interests include data mining, machine learning, and load forecasting.

EENJUN HWANG (Member, IEEE) received the B.S. and M.S. degrees in computer engineering from Seoul National University, Seoul, South Korea, in 1988 and 1990, respectively, and the Ph.D. degree in computer science from the University of Maryland, College Park, in 1998. From September 1999 to August 2004, he was with the Graduate School of Information and Communication, Ajou University, Suwon, South Korea. He is currently a Faculty Member with the School of Electrical Engineering, Korea University, Seoul. His current research interests include database, multimedia systems, information retrieval, big data processing, and healthcare applications.